Signal Estimation in Structured Nonlinear Systems with Unknown Functions

by

Eric Leon Wemhoff

B.S. (California Institute of Technology) 1994
M.S. (University of California at Berkeley) 1998

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering-Mechanical Engineering

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Andrew K. Packard, Chair
Professor Kameshwar Poolla
Professor Laurent El Ghaoui

Spring 2003

Signal Estimation in Structured Nonlinear Systems with Unknown Functions

by

Eric Leon Wemhoff

Abstract

Signal Estimation in Structured Nonlinear Systems with Unknown Functions

by

Eric Leon Wemhoff

Doctor of Philosophy in Engineering-Mechanical Engineering

University of California at Berkeley

Professor Andrew K. Packard, Chair

We consider the estimation of unknown signals in structured models that are interconnections of known linear dynamic systems and unknown static maps, and contain unmeasured exogenous disturbances. A main motivation for analyzing such an issue is a system identification problem in which such an interconnection exists, and the static maps are to be identified when the inputs and/or outputs of the maps themselves are not available, and instead we must investigate them by interacting with the larger system.

Our technique is to formulate criteria and then search for estimates of the unmeasured signals based on three main types of criteria, these being that they are consistent with the linear dynamic system, that stochastic assumptions for disturbance processes are met, and that input-output pairs of the static maps are consistent with there being a static relationship between them. After revealing the basic approach to estimating signals, some time is spent on each of these three main parts of the estimation problem, presenting alternatives and implementation details.

The "staticness" consideration is a main contribution, and we present various possibilities for enforcing it. These are what make our formulation different from some other common estimation methods such as the Kalman filter or a least squares formulation.

Computational considerations are important because the entities being estimated are signals, and so the number of decision variables is necessarily large. We focus on solution elements which are compatible with efficient convex programming methods, and what can be done when they are not. We show examples and evaluate performance and usability of the method.

In a later chapter we present an approach to computing bounds on how good estimates can be guaranteed to be, for estimation formulations that meet certain assumptions. Alternatively, something can be learned from the bounding ideas about issues that make an estimation problem harder or easier.

Professor Andrew K. Packard
Dissertation Committee Chair

Contents

List of Figures **v**

List of Tables **vii**

Notation **viii**

1 Introduction **1**
1.1 Kernel Models . . . 3
1.2 Curse of Dimensionality . . . 4
1.3 DEQ Models . . . 6
1.4 Parameter Estimation: Strengths and Weaknesses . . . 6
1.5 Structured Models . . . 10
1.6 Outline . . . 13

2 Framework **16**
2.1 The General Discrete-Time Model . . . 16
2.2 Example of Structure . . . 18
2.3 Canonical Model Form . . . 20
2.4 Problem Statement . . . 25

3 Elements **29**
3.1 Approach . . . 29
3.2 Linear System Constraints . . . 34
3.2.1 Parametrization of the Feasible Set of the Linear System Constraint 34
3.2.2 Linear Dynamic Systems and Operators They Define . . . 42
3.2.3 Computing a Particular Solution . . . 46
3.2.4 Computing a Nullspace Basis . . . 56
3.3 Staticness Criteria, and Other Properties of $\mathcal{S}$ . . . 62
3.3.1 Dispersion Function . . . 68
3.3.2 Related Staticness/Smoothness Measures . . . 77
3.3.3 Extensions of Dispersion to Multi-Input Operators . . . 83
3.3.4 Lipschitz Smoothness . . . 86
3.3.5 Other Structure in $\mathcal{S}$ . . . 88
3.4 Stochastic Criteria . . . 90

3.4.1 Mean 91
3.4.2 Variance 93
3.4.3 Correlations 96

4 Formulations **102**
4.1 Assembling Programs 102
4.1.1 Review Of Elements 102
4.1.2 Fixed z vs. Free z 106
4.1.3 Multiple Experiments 108
4.2 Example: Bungee 110
4.3 Example: ECP 125

5 Error Bounds for Estimates of Static Functions **143**
5.1 Motivating Example and Main Ideas 144
5.2 Some Features of Bounds and Estimates 149
5.3 General Formulation 163
5.4 Review 170

6 Conclusions **172**

List of Figures

2.1 Dynamic system with state x 18
2.2 Schematic of simple mass-spring-friction system. 18
2.3 Canonical model structure 22
2.4 Canonical model set 26

3.1 Linear system constraints 35
3.2 Static operator 62
3.3 Two systems driven by the same input: looking at the scatter plot from a dynamic vs. a static system 64
3.4 Intuition for associating a scatter plot with a static relation gets better as length of data record increases. 65
3.5 Frequency response of noise-coloring filter for input in Monte Carlo simulation of dispersion function. 73
3.6 Operator $\mathcal{S}_2$. 73
3.7 Sample means (solid lines) and ± 1 sample standard deviation sleeve (dashed lines) of the dispersion $\mathcal{J}(z, \mathcal{S}_i z)$ with random inputs, for four operators. . . 74
3.8 Empirical distributions of the dispersion measure for a random input, for two data lengths and four operators $\mathcal{S}_i$. 75
3.9 Dispersion $\mathcal{J}(z, \mathcal{S}_i z)$ with sine chirp input, for four operators. 76
3.10 Scatter plot with outlier points. 79
3.11 Sample means (random inputs) vs record length L for 8 of the dispersion-like measures ($\mathcal{J}$, $\mathcal{J}_w$, $\mathcal{J}_{\mathrm{A}}$, $\mathcal{J}_{\mathrm{TV}}$, $\mathcal{J}_2$, $\mathcal{J}_{\mathrm{D1}}$, and $\mathcal{J}_{\mathrm{D2}}$), for four operators each. In this figure $\mathcal{J}_{\mathrm{D1}}$ and $\mathcal{J}_{\mathrm{D2}}$ are divided by 1000 and 100000, respectively. . . . 82
3.12 Construction of two-input dispersion function. 84

4.1 Typical simulation data for the bungee example. 111
4.2 Estimated e, w, and z signals using 400 points, for three noise levels. 112
4.3 Scatter plots of estimated w and z, for three noise levels and different amounts of data. 113
4.4 114
4.5 Five of the bootstrap iterates. $\sigma_e = 0.2$, $L = 800$ 114
4.6 Four iterates from each of seven bootstrap sequences starting from random $f^{(0)}$, and one starting from exact simulation signals. 116

4.7 Scatter plots of estimates using various Γ_z. 117
4.8 Scatter plots of estimates using various Γ_e. 118
4.9 Correlation functions of the estimates of Figure 4.8, for various Γ_e. 119
4.10 Alternative ways to compute estimates. 120
4.11 Alternative ways to compute estimates–signals. 121
4.12 Estimation formulations using various staticness measures. 124
4.13 ECP "Industrial Emulator" controls experiment. 126
4.14 Simulation data for ECP noload model. 129
4.15 Trials for estimation of simulated Coulomb friction nonlinearity in ECP noload model. 131
4.16 Excitation of ECP drive system during 9 chirp experiments. 133
4.17 Varying Γ_z and σ_e. 134
4.18 Estimates of Ψ_*, a piecewise linear fit, and simulations. 136
4.19 Simulation data for ECP model with load inertia. 139
4.20 Effect of the weighting on $\mathcal{S}_2$ in the dispersion function. 141
4.21 Effect of the weighting on $\mathcal{S}_2$ in the dispersion function. 142

5.1 Example 1: Input data records z^A and z^B. 152
5.2 Example 1: $m_1(z_a, z_b)$ bound for z^A, uniformly distributed. 153
5.3 Example 1: $m_1(z_a, z_b)$ bound for z^B, normally distributed. 153
5.4 Example 1: Cross-sections of pairwise bound functions using z^A and z^B. . . 155
5.5 Example 2: The four inputs. 157
5.6 Example 2: Bounds and formulation A estimates of $\mathcal{S}_1$, for the four inputs. 158
5.7 Example 2: $\mathcal{S}_1$ and $\mathcal{S}_2$. 159
5.8 Example 2: Bounds and formulation A estimates of $\mathcal{S}_2$, for the four inputs. 160
5.9 Example 2: Bounds and formulation B estimates of $\mathcal{S}_1$, for the four inputs. 162
5.10 Example 3: Bounds and formulation A estimates of $\mathcal{S}_1$ and $\mathcal{S}_2$, for three different $G(0)$. 164
5.11 Example 3: Bounds and formulation B estimates of $\mathcal{S}_1$ and $\mathcal{S}_2$, for three different $G(0)$. 165

List of Tables

4.1 A collection of elements for use in formulating estimation problems. 103
4.2 Dispersion objective for bootstrap iterates ($L = 800$, $\sigma_e = 0.2$) 115
4.3 Dispersion objective for bootstrap iterates ($L = 800$, $\alpha_e^2 = 32.4044$, simulation initial conditions) 116
4.4 Constants in the ECP model. 127
4.5 Data records of estimates in Figure 4.18 135

Notation

$\mathcal{R}(A)$	range space of a matrix A
$\mathcal{N}(A)$	null space of a matrix A
$\dim \mathcal{X}$	dimension of the linear space $\mathcal{X}$
$\text{rank}(A)$	rank of the matrix A
$A \succ (\succeq, \prec, \preceq)\ 0$	For A a symmetric matrix, it is hermitian, positive definate (positive semidefinate, negative definate, negative semidefinate); for A a vector, it is componentwise positive (etc).
$A \succ B$	means $(A - B) \succ 0$
$x./y$	element-by-element division of the elements in the vectors x and y
L	length of a data record
$G \sim \left[\begin{array}{c\|c} A & B \\ \hline C & D \end{array}\right]$	G is a linear system with state-space data (A, B, C, D)
$\mathcal{L}$	the linear part of the canonical model
$\mathcal{S}$	the static part of the canonical model
z	input signal of the static part
w	output signal of the static part
$\mathcal{L} \circledast \mathcal{S}$	LFT interconnection of $\mathcal{L}$ over $\mathcal{S}$
n_v	dimension of the signal v
$v_{[t,s]}$	the collection or vector $(v(t), v(t+1), \ldots, v(s))$
$\overline{v}$	sample mean (average) of a vector v (page 91)
$\mathbf{Co}(X)$	convex hull of a set of points X
$X \sim F(x)$	X is a random variable with probability distribution F
$X \sim \mathcal{N}(m, \sigma^2)$	X is a normal random variable with mean m and variance σ^2
$X \sim U[a, b]$	X is a random variable whose distribution is uniform on $[a, b]$
$X_n \xrightarrow{\text{d}} D(x)$	the sequence of random variables X_n converges in distribution to a

	random variable with distribution D
z_α	α'th percential of the normal distribution
$f \in L_\gamma$	the function f is a Lipschitz function, with Lipschitz constant γ (page 86)
$z_{[k]}$	the subset of the n_z scalar signals comprising the input z to $\mathcal{S}$ that the k'th single-output function $\mathcal{S}_k$ depends on (see e.g. page 89)

Acronyms

IID	independent, identically distributed
WLOG	without loss of generality
LTI	linear time-invariant
LTV	linear time-varying
SISO	single-input single-output
MISO	multi-input single-output
MIMO	multi-input multi-output
LFT	linear fractional transformation
CAL	university of california at berkeley
ETFE	empirical transfer function estimate

Acknowledgements

There are many people to thank who contributed to this thesis, helped me make it through challenging periods, and made life enjoyable during my time at Cal.

First and foremost I owe Professor Andy Packard, my thesis advisor, a debt of gratitude, for his patience and support over these many years. He provided a superb environment in which to learn and explore at my own pace, and endless good will. Not to mention the willingness to discuss at length with me many of the smaller details when I needed another point of view. Maybe this is why I stuck around so long.

I want to thank Professor Kameshwar Poolla for his help and original ideas, and for introducing me to the area of system identification and teaching me much of what I know. I got lucky in that in the BCCI lab, you get two great advisors for the price of one.

Thanks also goes to Professor Laurent El Ghaoui for his instruction in the ways of optimization, his understanding, and unfailing interest in how I was progressing. I should not forget some of my first experiences at Berkeley working with Professor Gary Chapman; he helped me get off to a good start here with solid support and frequent discussions in his office. I would also like to take this opportunity to thank Professor Richard Murray, who more than anyone else initially got me excited about this field as I worked in his lab as an undergrad.

And there are many others. I was lucky to share the lab with many wonderful students in the BCCI lab and I greatly enjoyed working with all of you, both past and present, without exception. The same goes for many of the students from other groups in the mechanical engineering department, and the rest of the control faculty here as well. I always looked forward to working in the SPACE lab at Cal State L.A., and I could always count on my cycling pals to help me work off some steam.

Most importantly I want to thank my parents and my sisters for their continuing love and support during these many years. It's done.

Chapter 1

Introduction

This dissertation is concerned with system identification, for models that are structured and nonlinear. The types of structure that we consider consist of interconnections of linear dynamic and nonlinear static components. We propose identification criteria that can be used to formulate tractable optimization problems geared toward forming estimates of static nonlinear components in a nonparametric fashion. Attention is given to practical details in formulating and implementing these criteria so that the reader might quickly be able to try some of the ideas.

In virtually every field of study models are employed to aid in representing and understanding complex relationships. Models can help to to capture the dependences between variables in the problem, predict future outcomes, or understand the mechanisms involved in a process. Given some system of interest and some task, an important question is how to come upon a model suitable to the task.

There exist attractive models for many systems of interest, especially simple ones. In 'modeling' one endeavors to break up a complex system into smaller components, apply existing models for each of the components, and interconnect these to form a model of the whole. There are various occasions when more is needed.

- Some parts of the problem are not well understood. Then empirical modeling is used to find good models, and possibly give insight into the process being studied.
- There may be unknown parameters in the model, or incorrect assumptions regarding them. In this case parameters need to be identified or calibrated.
- First principle models can be very very complex; empirical modeling can be used to

find simpler, workable approximations.

System identification, or empirical modeling, is the process of combining prior knowledge, desired features, and quantitative observations of a device of phenomenon, to find a model that is a suitable representation for the task at hand. "Suitable" is generally some combination of being of a form and complexity that makes the model convenient to work with, as well as having sufficient fidelity to help solve problems of interest.

We are especially concerned here with identifying dynamic models, for which there is a notion of time, and the dependent variables (the *inputs*) at a given time can depend on the independent variables (the *outputs*) at that time as well as previous times. This is the primary focus of "system identification", with static modeling being an important subset of problems. The problem has been studied in association with varied subjects, but especially within the fields of controls and statistics. Further introduction to system identification (with the primary emphasis on linear modeling) can be found in Johansson [22] or Ljung [25]. The subject is important given the vast array of situations in which mathematical models are employed, and is challenging, both mathematically and philosophically.

It is useful to keep in mind a common generic description of the system identification process. Usually, the procedure can be broadly broken down into these parts. Start with a device, system, or phenomenon $\mathcal{A}$ which we desire to model. Specify a *model set* $\mathfrak{M}$ of models from which we expect it is possible to find a satisfactory one, along with a criterion for choosing a "best" element $\mathcal{M} \in \mathfrak{M}$ which is some sort of assemblage of the various assumptions and constraints and desires that we want our model to satisfy. At this point choosing the model is, in theory, a rather mechanical process of evaluating each candidate model from the model set according to the optimality criterion. Of course for most formulations we might try this process does not turn out to be so simple, and it's the search for tractable procedures that guides research in the system identification field.

When choosing from a model set that is a collection of linear dynamic systems, system identification is well developed. Although research does continue, the issues and tradeoffs are well understood and methods for linear system identification are mature. This includes the ability to handle parametric estimation (e.g. prediction error parameter estimation methods) and nonparametric estimation (e.g. empirical transfer function), frequency and time domain methods, models with structure as well as canonical models, and uncertain models Ljung [25], Söderström and Stoica [39], Johansson [22]. Further, there are computational tools out there which handle a wide array of linear system identification problems

Ljung [26].

However, as would be expected for the much more complex case of nonlinear systems, results are more difficult, more fragmented, and less general. The need arises when the class of linear models is too restrictive. Many physical systems exhibit some degree of nonlinearity. Often it's sufficient to represent a system using a linear model, but the nature of the nonlinearity, applications requiring high-fidelity modeling, and the desire to increase performance over that afforded by linear analysis, often make it necessary to look to nonlinear model sets. For instance in controls applications a linearization of the plant behavior about an operating point can be sufficient for regulation problems, but problems involving tracking or path planning where there are large deviations may make it more important to account for nonlinearness. If the system has non-smooth linearities at the origin (e.g. Coulomb friction) then linearization may be inappropriate.

A brief overview of methods in nonlinear identification is in order. To orient oneself a number of fairly recent surveys covering various aspects of the subject may be useful, including Billings (1980), Sjöberg et al. (1995), Juditsky et al. (1995), Pearson (1995), and Haber and Keviczky (1978).

1.1 Kernel Models

Some approaches begin with a very general model set. A large class of nonlinear functionals can be represented in the form of a Volterra series, which maps input signals u to output signals y as

$$y(t) = \sum_{n=0}^{\infty} \int \dots \int h_n(\tau_1, \dots, \tau_n) \prod_{i=1}^{n} u(t - \tau_i)\, d\tau. \tag{1.1}$$

The behavior of the model depends on the kernels $h_n(\cdot)$ of the integral functionals, and it is these functions that are to be identified. A closely related model was introduced by Wiener, with the form

$$y(t) = \sum_{n=0}^{\infty} [G_n(k_n, u)](t), \tag{1.2}$$

where the functionals G_n are also integral equations in u, with kernels $k_n(\tau_1, \dots, \tau_n)$ (these were used by Wiener as a sort of orthogonalized version of the Volterra kernels, via a Gram-Schmidt procedure). In discrete-time models the integrals are replaced by sums.

Much of the earlier studies in nonlinear system identification focused on estimating the kernels in these models. They are an extension of the convolution representation of linear systems, as in

$$y(t) = \int H(t,\tau)u(\tau)\,d\tau. \tag{1.3}$$

Here the kernel H is the impulse response. For linear time-invariant (LTI) systems a related model description is the transfer function, the Laplace transform of $H(t-\tau)$. Analogously we can consider the transform domain for Volterra kernels, using the multi-dimensional Laplace transform, for nonlinear time-invariant systems. Volterra models are also an extension of a polynomial expansion for static functions. For instance if the kernels in (1.1) are of the form $h_n(\tau_1,\ldots,\tau_n) = \bar{h}_n\delta(\tau_1,\ldots,\tau_n)$ (multidimensional Dirac delta function) then the model reduces to

$$y(t) = \sum_{n=0}^{\infty} \bar{h}_n u^n(t). \tag{1.4}$$

Schetzen (1980) goes into more depth about the theory of Volterra and Wiener functionals. An important result is that every continuous nonlinear time-invariant operator can be approximated arbitrarily closely by a Volterra functional (Frechet 1910, Boyd and Chua 1985). The surveys by Hung and Stark (1977) and Billings (1980) discuss available identification methods for the kernels. These generally treat the input and output as stochastic processes, and work in terms of estimates based on joint correlation and higher-order statistics of the output and lagged versions of the input. Some provide an estimator for the value of a kernel at a point in its input space e.g. $\hat{k}_n(\tau_1,\ldots,\tau_n)$, others work with various functional expansions of the basis functions and derive estimators for the expansion coefficients (the former lead to nonparametric methods and the latter, parametric). The computation for the estimates is involved, and usually assumptions are made on the distribution of u which make the problem more solvable (typically white, Gaussian statistics). Others have worked in the frequency domain, producing estimates that are analogous to the empirical transfer function estimate (ETFE) of LTI systems (Brillinger [7], French and Butz [13]).

1.2 Curse of Dimensionality

An important fundamental difficulty arises, even when computation of these estimates is doable, commonly called the *curse of dimensionality*. Suppose we were able to

somehow "sample" the n'th kernel function $h_n(\tau_1, \ldots, \tau_n)$ directly at any τ we wished. This is of course an ideal scenario, for with a typical system there is no possibility to isolate h_n in this way. But suppose it is possible. To gather data about h_n on a grid with L points per input dimension, or to gather information about h_n in some other representation (e.g. a function expansion) but with the same granularity, or resolution, requires on the order of L^n samples. This says something about how much data will be needed, even in an ideal situation, to identify h_n to a desired accuracy. The task grows exponentially with n.

Indeed, even the problem of how to efficiently *represent* functions of more than two or three variables, or store them in computers, is nontrivial. It implies that, without further knowledge of h_n, for large kernel order n (hence large input dimension) the amount of data needed to achieve accurate results quickly becomes unmanageable.

In the case of kernel models the problem is compounded by their moving-average nature. Think about finite-dimensional stable LTI systems. The impulse response is a completely general representation, but often it's very inefficient, in particular for systems with infinite impulse response. We could approximate the system by retaining only those points which are greater than some threshold, and the shorter the system's "memory" i.e. the faster the impulse response dies out, the less we have to store. But in general even this will require much more than the small number of parameters in a state space or input-output difference equation representation. In the same way for the Volterra series, the portion of the domain of $h_n(\tau_1, \ldots, \tau_n)$ over which we need to measure and store it for a given level of approximation is determined by the length of the system's memory. This is bad news for using kernel models to represent autoregressive nonlinear dynamic systems.

The strength of kernel models is their generality. However they tend not to perform the representation elegantly, and due to data and computational requirements authors very rarely have attempted to estimate more than up to the second or third order kernels (Billings (1980)). In this case the dream of a model set that can fit essentially anything becomes a rather limited functional space. A further disadvantage of these models is the difficulty of incorporating *a priori* knowledge, or in giving any sort of interpretation to estimates (Hung et al. 1977).

1.3 DEQ Models

The other type of model that has been used in nonlinear identification is represented by differential or difference equations (DEQs). The general nonlinear state-space model, in discrete time, is

$$x(t+1) = F_x(x(t), u(t), e(t)) \tag{1.5a}$$

$$y(t) = F_y(x(t), u(t), e(t)), \tag{1.5b}$$

and the prototypical input-output model that has been studied is the NARMAX (non-linear auto-regressive moving-average exogenous input)

$$y(t+1) = F(\phi(t)), \tag{1.6}$$

where $\phi(t) \in \mathbf{R}^n$ is a regressor vector of selected past inputs, outputs, and disturbances,

$$\phi(t) = \big(y(t), \ldots, y(t-n_y), u(t), \ldots, u(t-n_u), e(t), \ldots, e(t-n_e)\big). \tag{1.7}$$

These include as special cases most of the DEQ models sets that have been studied (Chen and Billings 1989). DEQ models commonly include the noise process e which accounts for unmeasured disturbances and mismodeling, which is not commonly found in kernel representations.

Here it's typical to fix the order n_x for the state-space model, or n_y, n_u, and n_e in the NARMAX, which defines a space $\mathcal{U}$ of functions, and the model set associated with e.g. (1.6) is generated as F varies over $\mathcal{U}$. Sometimes we limit $\mathcal{U}$ to contain only continuous functions, or ones having other smoothness properties. With these model sets the identification task is to choose the optimal element from an infinite-dimensional function space, based on a performance criteria.

1.4 Parameter Estimation: Strengths and Weaknesses

The most general tools available for solving problems that arise in system identification are derived from generic nonlinear programming (NP) methods which search for an optimal parameter vector that minimizes an objective function. This usually involves a model set that is specified in terms of a finite set of unknown parameters and possibly constraints on the parameters, and a criteria that measures the quality of a particular value

of the parameter, typically in terms of the corresponding model's ability to match measured input-output data. Depending on the parametrization and objective the solution will be more or less computationally attractive. For instance oftentimes the problem is formulated so that in the end it is a least squares problem, which is easily handled. For more general criteria and constraints, assuming they are differentiable, a variant of Gauss-Newton descent is a popular choice.

In a stochastic setting this process is known as *parameter estimation*, where we more or less treat the model as making probabilistic statements about outputs given the regressor vector, and search for the parameters that make the model most consistent with the distribution of the unknown disturbance and observed input-output statistics. A typical performance criteria for this is the sum of squared prediction errors. This criteria is attractive in terms of estimator properties and convergence analysis, and works well in practice.

Optimization options are a good fit for situations in which there is enough prior knowledge to feel confident in restricting the search to a model set which is characterized by a small (finite) set of unknown parameters. This is known as a parametric model, and applying parameter estimation techniques we can achieve the dual objective of low bias and variance. Typically this type of model is the result of first-principles, physical modeling, and the parameters have natural physical meaning. It can also result from truncated function expansions.

In order to apply parameter estimation to searching infinite-dimensional spaces, like when there is an unknown function to be estimated, we must first restrict the search to a finite-dimensional manifold $\mathcal{V}_M$ of the original space $\mathcal{V}$, parametrized by a vector $\theta \in \mathbf{R}^M$. For instance it is common to expand the model function as a polynomial, $F(\phi;\theta) = \sum_{|\alpha|=0}^{\infty} \theta_\alpha \phi_1^{\alpha_1} \dots \phi_n^{\alpha_n}$ (using multiindex notation), and then truncate the number of terms used to a finite value.

An extremely attractive feature of the NP or parameter estimation approach is that it's a flexible tool. It can be made to work for a vast array of model structures, identification objectives, and constraints, and generally does not depend on special input signals. For instance it can be used for identification of the general DEQ models as well as Volterra and Wiener kernels or finitely-parametrized models. If there is anything resembling a big hammer in system identification (linear as well as nonlinear), this is it. For more background material on parameter estimation see Ljung (1999) or Sjöberg et al. (1995); for

nonlinear programming see Luenberger (1984) or Gill et al. (1981).

However with estimation of objects like F or h_n where the model set is essentially a function space, the choice of finite parametrization can be tricky. The goal is to locate the element of $\mathcal{V}$ that optimizes the performance objective. But in the parametrization if we use the wrong "basis" functions then $\mathcal{V}_M$ will not contain this element, nor a good approximation to it, and the actual estimate will be of poor quality. Other problems arise if the parametrization is too large. Solving the nonlinear optimization problem becomes harder, the computation involved in searching a large dimensional space becomes prohibitive. Local minima can be a problem as well, and so choosing an intelligent starting point is crucial.

A more fundamental reason not to simply use an extremely general function expansion is known as the bias/variance tradeoff (also called the parsimony principle or overfitting problem). In the stochastic setting once again, consider a probabilistic experiment where an input-output record of length N is generated according to a "true" model F_0, we use a prediction error performance criteria to pick an optimal parameter estimate $\hat{\theta}_{M,N} \in \mathcal{V}_M$ (of size M) based on the record, then another data record is generated, and we compute the squared prediction error, $V_{M,N}$. This quantity is a random variable which measures the quality of the models generated as we repeat this experiment. Under some standard assumptions it is straightforward to show that the expected error $E[V_{M,N}]$ can be attributed partly to *bias* and partly to *variance* (Juditsky et al. 1995, Ljung 1999). Let $\theta_{M,*}$ be the model in $\mathcal{V}_M$ which has minimum expected squared error; the error of this model is the bias part. If $\mathcal{V}_M$ includes a model which exactly matches the data-generating system, then the bias is zero, and hence this seems to imply we should choose M as large as our computer can handle. However since the data record is of finite length and noisy, the estimate $\hat{\theta}_{M,N}$ deviates randomly from $\theta_{M,*}$, and this deviation is responsible for the variance part of $V_{M,N}$. This part of the error is sensitive to the number of parameters; as the record length N becomes larger or the noise level decreases it becomes smaller, however as M increases it becomes larger, as the estimate becomes more sensitive to noise in the data.

Therefore when using function expansions in order to apply parameter estimation tools, one needs to choose basis functions wisely, in order to locate a quality estimate (small bias) with a small number of parameters (small variance). For functions with high input dimension this is especially difficult since the size of the function expansion grows exponentially with the dimension. Ideally we want to have an expansion that 'matches the system', and if we don't know how to do that, one that is somehow known to work

well in general. Several standard choices have been studied, known under various names. Polynomials, neural networks, radial basis functions, wavelets, splines, hinging hyperplanes, fuzzy logic. An excellent discussion of the various choices and their properties can be found in the two-part survey, Sjöberg et al. (1995) and Juditsky et al. (1995).

Besides imposing a finite parametrization and applying parameter estimation, other tools for dealing with the challenges of infinite-dimensional model elements include regularization and nonparametric methods.

In regularization methods the approach for unknown functions is to go ahead and use a large function expansion to parametrize the function space, and then during the optimization phase add a term that penalizes the model complexity [38], [21]. In this way the number of terms in the expansion that actually get used, and hence the variance contribution in the expected square error, can be controlled, while retaining the flexibility of the more complete parametrization of the space. The regularization penalty forces the solution to use the "right" basis functions. An attractive feature is it is compatible with parameter optimization schemes. The practical problems of large nonlinear programming problems remain. And the possibility remains of not including appropriate basis functions. For instance radial basis functions are a common choice for general function approximation, but they are inappropriate if the function being approximated is linear. Regularization can't help if the right basis functions aren't included to begin with.

Nonparametric methods, on the other hand, tend to not exactly fit into the prototypical identification procedure described above, where choosing a model set and gathering data are independent steps. The empirical transfer function estimate (ETFE) for the transfer function of a linear system is an example of a nonparametric method. Another example is *nonparametric regression estimation* from statistics, where estimates of probability density functions are formed by smoothing observed realizations of the random variable under repeated trials. Roughly speaking, in methods given this label the philosophy is to be guided by the data. The form of unknown model elements isn't artificially restricted to a finitely-parametrized function space ahead of time, before data is gathered, and the spirit is to be able to potentially settle upon any element of the space. A common feature of nonparametric methods is that the precision with which a function is approximated depends on the length of the data record. As the data increases the approximation resolution increases, while for small records the approximation is rough. In this way a certain level of smoothing, or rate of data compression, is maintained. A survey paper by Härdle et al.

(1997) points out this and other characteristic features of nonparametric methods and discusses some common approaches that are seen in statistics and time series analysis. They state "[nonparametric methods] are ... an exploratory tool for a better understanding of the underlying dynamics of the process and a starting point for finding more parsimonious models." Oftentimes a nonparametric method is essentially a way to transform data, in order to aid in understanding and insight into the potential structure in a problem, for instance as an aid in choosing a block diagram structure, or efficient function expansions for parameter estimation methods.

1.5 Structured Models

The model sets represented by the class of kernel models (1.1) and (1.2), or the classes of DEQ models (1.5) and (1.6), are attractive for their generality. In principle they can represent, or approximate closely, almost any model we might consider. They are used in *blackbox* identification situations, in which no use is made of *a priori* knowledge about the system to be identified, and so the estimate is based solely on input-output data. Little or nothing is assumed about the device under test, and so the model set must be general enough to be ready for anything.

It's tempting to argue that since methods exist for working with these model sets, then using these tools we can solve almost any identification problem. In reality there are a number of reasons to use a more structured, i.e. a smaller, more constrained, model set.

- **apriori information**: Often there is a reason to suspect that a model with a specific form will perform well in representing a system. In this case one wants to take advantage of this insight to restrict the set of models under consideration. Usually this type of knowledge would be motivated by physical insight or modeling, or by a preliminary "structure identification" step in the system identification process.
- **curse of dimensionality**: To some extent structured models can reduce the curse of dimensionality, by replacing a single unknown function with large input dimension with unknown parameters and functions of smaller input dimension.
- **structure identification**: Sometimes one wants to test the relative ability of various model structures to capture a certain behavior. In this case one may not know if a

system can be well represented by a model with a particular structure, short of trying it. Sometimes an additional outer loop is used to determine the best model set.

– **end use**: The intended use of the model may impose restrictions on what model structure we are willing to accept. For instance a linear plant model is needed if one intends to apply linear model-based control design methods. A main motivation for modeling in the first place is to provide a simple, transparent, and manageable version of reality, or at least the part of it one is aiming to understand, predict, or control. This tends to be at odds with Volterra and Wiener kernel models, as well as DEQ models where F is the result of a function expansion in terms of a huge number of nonphysical basis functions and parameters. These models can be unwieldy and hard to interpret or relate to more physical characteristics. For control design simple difference or differential equation models are most amenable to control system design and analysis. Sometimes the goal is simply to gain understanding and insight into the operations of the system, and the goal is essentially to find a simple or elegant model that can reproduce the system's behavior. For this reason it may be desirable to constrain the model to be an interconnection of relatively simple components.

– **subsystems**: A model is sought for a physical subsystem which is not easily experimented on independently of a larger system; thus it must be possible to decompose an identified model of the larger system into a connection of elements, some of which are the subsystems of interest.

"Structure" can mean a lot of things. For instance we could constrain the model to be stable, or to possess other qualitative characteristics, or to have a specific steady state gain, or unknown functions to possess certain smoothness properties. For our purposes a *structured* model set is one that is smaller or more constrained that the general models (1.5) or (1.6), and we're usually talking about the form of the model equations and the interconnections that define them when we talk about structure.

In terms of this meaning of structure, a sort of middle ground between parametric model sets (which are highly structured) and the extremely general generic forms, are ones formed by interconnecting simpler model sets. We can picture it graphically as a block diagram, where the model for a given block is chosen from a specified class. Possibly this choice of structure is guided by *a priori* information, and possibly the individual components have some physical meaning. The task now is to pick the appropriate element from each of

these constituent model sets. The hope is the individual model sets are more manageable, being either parametric, or possibly a general form such as a NARMAX model set but where the state dimension is smaller than the overall model set, or possibly a static function but with smaller input dimension than unknown functions involved in a more general model for the overall system. The overall model is constrained by the structure, and has a less general form than e.g. a state-space model of the same overall state, input, and output dimensions. While being more constrained and thereby to some extent alleviating curse of dimensionality and variance issues, these model sets endeavor to retain some of the universal nature of the more general sets mentioned above, in that they might be successfully applied to many different situations.

Along these lines, a class of model that has been studied extensively are the so-called *block-oriented* model structures. These are interconnections consisting of linear time-invariant dynamic systems and static nonlinear functions. The overwhelmingly most common case is the cascade or "sandwich" models (linear dynamic followed by static nonlinear followed by linear dynamic), or the two special cases of this known as the Wiener model (static nonlinear followed by linear dynamic) and Hammerstein model (linear dynamic followed by static nonlinear), although other possibilities are occasionally studied. The added structure in the model equations allows a particular model in the set to be represented with a much smaller amount of information. For instance a SISO Hammerstein model involves two model sets, one a SISO static mapping, the other a SISO LTI system. The LTI can be represented generally by a single-input function (its impulse response), or if a model order is chosen, a finite set of parameters, e.g. difference equation coefficients. In general linear dynamic systems have a much much more compact representation than the general nonlinear dynamic system of the same order. In this case the identification problem has been reduced to searching two single-input function spaces, or one single-input along with a finite parameter vector. Compare this to the large input dimension of the unknown function in (1.6).

Many specialized identification methods exist for block-oriented models. The vast majority of studies have considered SISO elements only. Parameter estimation techniques can be readily applied once a parametrization is chosen for the LTI and static components. This involves the advantages and disadvantages discussed previously. Chen and Fassois (1992) considered sandwich models, using a polynomial basis expansion for the static element and input/output difference equation representation of the discrete-time LTI elements,

a prediction error criterion, and used Gauss-Newton descent. Studies which parametrize the model and apply iterative schemes that use least squares in an intermediate step include those of Lacy and Bernstein (2002), for Wiener models, and Narendra and Gallman (1966), for Hammerstein models. Many have applied nonparametric correlation and frequency domain techniques that were developed for kernel models to form nonparametric estimates of the LTI parts (Billings 1980). Vandersteen and Schoukens (1997) consider a model set which is a feedback interconnection of 4 LTI system and one well-behaved static function. They use special input signals to form estimates of the first three Volterra kernels which can then be used to solve for the frequency responses of the LTI systems. Billings and Fakhouri (1982) shows that so-called *separable* inputs can be used to achieve a similar objective of identifying the first two Volterra kernels and solving for the impulse response of the linear systems, independently of the static nonlinearity, using correlation analysis. The cases this method can handle include Wiener, Hammerstein, and feedback systems. Subspace identification has been applied in another, more recent method for identifying the LTI part independent of the nonlinearity, for Wiener and Hammerstein systems; see e.g. Westwick and Verhaegen (1996). These have the unique advantage of naturally handling MIMO linear and nonlinearities; however restrictive assumptions about the nature of the nonlinearity are required. In these latter studies, estimation of the nonlinearity is presented as an afterthought, once the linear parts are identified. On the other hand the static nonlinearity is the focus of a multitude of papers which use a nonparametric kernel regression estimate of the linear parts in Hammerstein systems which is independent of the nonlinearities, see e.g. Greblicki and Pawlak (1994). These require user choices of windowing functions and smoothing parameters.

1.6 Outline

This thesis introduces methods, nonparametric in nature, to treat block model structures involving known linear dynamic parts and unknown static functions that are to be identified. The hope, for these model sets, is to provide an alternative to parameter estimation which offers a more natural way of approaching the problem, one which does not involve choosing a finite function expansion or initializing parameter. Another goal is to be more general, in various ways, than many of the existing specialized methods for block oriented models. In terms of model structure we consider arbitrary feedback

interconnections of constituent components that are multi-input multi-output. There is no restriction of the form of the static nonlinearities, including discontinuous and noninvertible ones; several of the Wiener methods only work for invertible nonlinearities, and kernel model sets only contain analytic nonlinear functionals. We do not depend on using special inputs (although for achieving good results, some input signals are better than others). Assumed stochastic models for unknown disturbance signals are treated explicitly.

In Chapter 2 we define and discuss the class of models that are studied throughout this thesis. In Section 2.3 we introduce a useful canonical representation of the general model class, followed by a more precise statement in Section 2.4 of the specific identification problem that the majority of this dissertation is concerned with.

In Chapter 3 we present the constituent elements of a nonparametric identification methodology for the static nonlinearities. These elements use mainly three types of criteria and constraints: the model structure, including the linear dynamic part of the model set; the fact that the static functions are in fact static; and stochastic properties of unknown disturbance signals. Various motivations for and properties of these components are discussed. We place an emphasis on formulating problems that can actually be solved today.

In Chapter 4 we continue looking at these components and how they might be combined to produce system identification methods that apply in different situations that might be encountered. Important distinctions here include:

- which signals of the block interconnection are measured
- what are the sizes of the static functions i.e. what is their input dimension
- what apriori information do we have regarding smoothness properties, etc of the functions
- which properties of the stochastic disturbance process are important.

This chapter also contains examples demonstrating properties and performance of various assemblages of criteria and constraints, as these distinctions vary. Some consideration is given to how computationally tractable the proposed procedures are, and how friendly they are to implement from a user point of view.

For a certain class of the estimation formulations presented, a method to compute a sort of bound on the estimation error is discussed in Chapter 5. The bound depends on the assumed linear part of the system, Lipschitz assumptions about the unknown static functions, and the particular input to the system used in the data-generating experiment,

and are computational in nature. In addition to bounding the estimation error, the ideas can be used to provide insight on how the estimation performance is affected by different input signals and features of the particular model structure of the problem. Some computational examples are presented, and we also see that general trends and lessons learned carry over to estimation formulations that are related to, but not contained within, the class for which the bounds are valid.

Chapter 2

Framework

The class of models we will be using to describe system behavior, and the types of models that our system identification efforts result in, are state space representations of discrete-time dynamic systems. The discrete-time choice is natural in system identification because these problems are driven by input/output data which is typically gathered at discrete sampling instants (although identification of continuous-time nonlinear models is an important topic of contemporary interest, see e.g. Billings and Li [3] or the review paper of Unbehauen and Rao [40]).

2.1 The General Discrete-Time Model

The following system of first-order difference equations is a very general representation of a discrete-time state-space model:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), e(k), k) \\ y(k) &= h(x(k), u(k), e(k), k). \end{aligned} \tag{2.1}$$

Here y is the output, u and e are inputs, and x is the internal state of the system; f determines how the state transitions and h is the readout function. These functions are vector-valued in general, as are all signals.

For a given device or phenomenon it is generally an unattainable goal to find a model which allows perfect computation or prediction of the measured output of that system. A model is hard-pressed to account for every possible factor that might affect the system, the effects of the inputs it does account for are most likely imperfectly modeled,

and those inputs are themselves known only approximately.

In light of this it is common practice to introduce model elements, disturbances or uncertainties that cannot be known completely. We consider models are affected by unknown disturbance inputs; these represent that part of the system that is not explained exactly in the model. An important part of such a model is any characteristics that can help describe these disturbances. For instance even though the disturbance is known, there might be good reason to think that usually it is "small", which may in turn be interpreted to mean that the model is "pretty good". It is typical to use a stochastic description to characterize typical disturbance signals. Part of this description may be embedded in the model itself, allowing for a simpler characterization of the exogenous disturbance. In models such as (2.1) the unmeasured or unknown disturbance part of the exogenous input is labeled e, and the known part is labeled u.

We take a moment now to formalize a few points and define some notation. To start off, all signals and functions associated with systems and models are real-valued. The notion of a *signal* is important; this is a mapping from *time*, T, a subset of $\mathbf{R}$, to $\mathbf{R}^n$. We deal with discrete time, in which time is a countable subset of $\mathbf{R}$ and taken without loss of generality to be $T = 1, 2, \ldots$; thus signals are vector-valued sequences. The dimension of a signal v is denoted n_v, and $v(t) \in \mathbf{R}^{n_v}$ is the value of the signal v at time t. As shorthand notation the sequence $(v(t), v(t+1), \ldots, v(s))$ will be denoted by $v_{[t,s]}$, and $\mathbf{R}^n[T]$ is the space of signals of size n defined on the times included in T. When working with matrix notation it will sometimes be useful to associate $v_{[1,L]}$ with the $L \cdot n_v \times 1$ vector

$$v_{[1,L]} = \begin{pmatrix} v(1) \\ \vdots \\ v(s) \end{pmatrix} \in \mathbf{R}^{Ln_v}$$

The operator $\mathcal{D}$ defined by (2.1) maps an initial condition $x(t_1) = x_1$ and an input signal $u_{[t_1,t_f]}$ to a state trajectory $x_{[t_1,t_f]}$ and output signal $y_{[t_1,t_f]}$ $(t_f \geq t_1)$. In equations we write $y_{[t_1,t_f]} = \mathcal{D}(x_1, u_{[t_1,t_f]})$, and in pictures we draw this as in Figure 2.1.

The size of the state vector n_x is known as the order of the system. In a system identification problem, to some extent the choice of the size of the state and unmeasured disturbance signal is part of the system identification problem. Often there is a natural choice, or alternatively various choices may be considered to see which results in the most attractive model. At any rate we shall assume that a model order n_x and number of

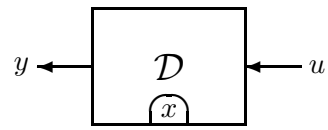


Figure 2.1: Dynamic system with state x

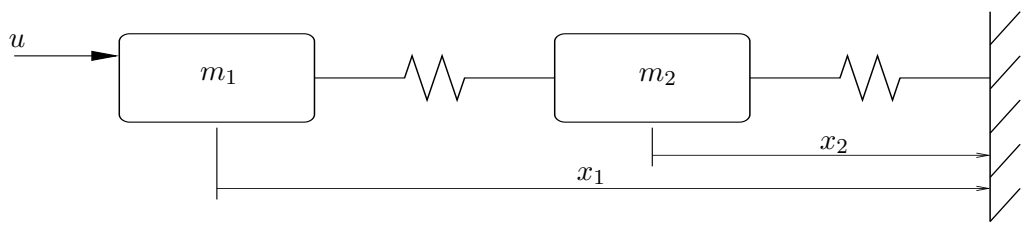


Figure 2.2: Schematic of simple mass-spring-friction system.

disturbance signals n_e have been settled on. Hence f and h have fixed input and output sizes.

Recall one part of any system identification problem formulation is the concept of a set of candidate models from which the "best" model is to be chosen. One possible model set can be formed by considering all systems that can be described by (2.1), that is any model which results from choosing functions f and h of the proper size. In this situation the system ID task is to identify those two maps, with no additional constraints on what those maps can be. This is often referred to as a type of *blackbox* model set, and correspondingly we label this set $\mathfrak{M}_{\mathbf{B}}$. As it stands this set is very broad, indeed for many specific problems of interest it is too large and we are really only interested in models that have a more restricted form.

2.2 Example of Structure

To make ideas more concrete consider a simple example system identification problem.

Example 2.1 Consider the mass-spring system diagrammed in Figure 2.2. This is the schematic of a physical system we wish to model. Masses 1 and 2 are connected by a spring, and mass 2 is connected by another spring to a rigid object. Each mass

experiences a frictional force acting on it. The input to the system which we are able to command is a force applied to mass 1.

For whatever reason we believe that a model that can accurately represent the system fits the following description. Each mass m_i has second-order dynamics between the resultant force acting on it and its position x_i. The springs' reaction forces are functions $k_i(\cdot)$ of their displacements, and possibly these functions are nonlinear. Similarly, we have reason to believe the friction force experienced by mass 1 is well described by a viscous force proportional to its velocity, while mass 2 experiences a friction force that is a nonlinear function $f_2(\cdot)$ of $\dot{x}_2$ such as Coulomb or Stribek friction.

In continuous-time this model is described by the differential equations

$$\begin{aligned} m_1\ddot{x}_1 + c_1\dot{x}_1 + k_1(x_1 - x_2) &= u \\ m_2\ddot{x}_2 + f_2(\dot{x}_2) + k_2(x_2) - k_1(x_1 - x_2) &= 0 \end{aligned}$$

or as a system of first-order equations,

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= \frac{1}{m_1}\Big[- c_1 x_3 - k_1(x_1 - x_2) + u\Big] \\ \dot{x}_4 &= \frac{1}{m_2}\Big[- f_2(x_4) - k_2(x_2) + k_1(x_1 - x_2)\Big]. \end{aligned}$$

In order to use a discrete-time model that approximates these dynamics, so that we can proceed with system identification, one possibility is to choose a sampling period T and approximate the derivatives with forward differences, i.e.

$$\dot{v}(kT) \approx \frac{v(k+1) - v(k)}{T}.$$

This gives

$$\begin{aligned} x_1(k+1) &= x_1(k) + Tx_3(k) \\ x_2(k+1) &= x_2(k) + Tx_4(k) \\ x_3(k+1) &= x_3(k) + \frac{T}{m_1}\Big[- c_1 x_3(k) - k_1(x_1(k) - x_2(k)) + u(k)\Big] \\ x_4(k+1) &= x_4(k) + \frac{T}{m_2}\Big[- f_2(x_4(k)) - k_2(x_2(k)) + k_1(x_1(k) - x_2(k))\Big]. \end{aligned} \tag{2.2}$$

The measured output of the system is the position of the second mass, x_2. There is additive output noise, e, hence the readout equation is

$$y(k) = x_2(k) + e(k). \tag{2.3}$$

We might for instance assume that e is white Gaussian noise with variance σ_e^2.

Good values to use for the parameters m_1, m_2, and c_1 are known, and T is determined by the rate at which data is sampled. The goal is to identify the functions $k_1(\cdot), k_2(\cdot)$, and $f_2(\cdot)$. This results in a candidate model set. Namely, we consider all models of the form (2.2), where k_1, k_2, and f_2 are free "parameters". The task is to find the "best" one of these models, and in the process the best estimates of the unknown functions, using input-output data and applying criteria for model fitness. This particular model set contains an implicit representation of prior knowledge about the internal workings of the system. We used it to construct the model set.

A slight twist on the problem is the following. Perhaps we have reason to believe that the two springs have identical properties, for instance two springs of the same manufacture were used to construct the device. In this case $k_1 = k_2 =: k$, and we have an unknown element of the model that is *repeated*. This reduces the model set further; in this case we only consider models with $k_1 = k_2$.

■

The model set in this example is one which has *structure*. The models included are precisely those whose model equations have the specific form described, and only those. It excludes many of the models in the blackbox model set associated with the general system (2.1), because we only consider those functions f and h indicated by (2.2) and (2.3). The individual components that comprise these functions and the specific way in which they are combined limit the possibilities for the system functions f and h.

2.3 Canonical Model Form

From this point onward we wish to work with models that are made up of two types of operators: linear (possibly dynamic) ones, and static (possibly nonlinear) ones. We consider interconnections of such, meaning that the input of a given component operator is a subset of the signals which are either external inputs to the overall model or outputs

of the component operators, and the output of the model is a subset of the outputs of the component operators.

By way of fixing notation, a linear time-varying dynamic system $\mathcal{G}$ with input u and output y is defined by its state-space equations

$$\begin{aligned} x(t+1) &= A(t)x(t) + B(t)u(t) \\ y(t) &= C(t)x(t) + D(t)u(t). \end{aligned}$$

We follow the common convention of using the matrix of packed state-space data to represent the system:

$$\mathcal{G} = \left[\begin{array}{c|c} A(t) & B(t) \\ \hline C(t) & D(t) \end{array}\right].$$

For dynamic operators in general, the output at time t depends on the the values of the input at all times $t \in T$. Static operators, however, have the following property.

Definition 2.2 (static operator) *An operator $\mathcal{S} : \mathbf{R}^m[T] \to \mathbf{R}^n[T]$ is static if there is a function $s : \mathbf{R}^m \to \mathbf{R}^n$ such that for all signals $z \in \mathbf{R}^m[T]$ and all $t \in T$,*

$$(\mathcal{S}z)(t) = s(z(t)).$$

That is the output of a static operator at a given time t depends only on the value of the input at time t, and not on any other portion of the input signal, nor on time explicitly.

Aside: for a static operator $\mathcal{S}$ a distinction might be made between the operator itself, which maps the input *sequence* z to the output *sequence* w, and the associated operator s which defines how the output at time t depends on the input at time t. The intended meaning is usually clear from context and in the sequel one symbol, such as $\mathcal{S}$, will often be used to refer to both.

A sort of canonical representation for models that are interconnects of linear and static components is achieved by grouping the linear operators into a larger linear operator called $\mathcal{L}$ and the static ones into a larger static one called $\mathcal{S}$. Thus we consider models which are an interconnection of a linear system $\mathcal{L}$ and a static one $\mathcal{S}$, as pictured in Figure 2.3. In addition to the model's external input and output, two new signals signals have been defined in this interconnection; z and w are internal signals which represent the input and output, respectively, to the static block. They are defined by the nature of the interconnection between $\mathcal{L}$ and $\mathcal{S}$. x is the state of $\mathcal{L}$, and as before u labels the inputs to the model

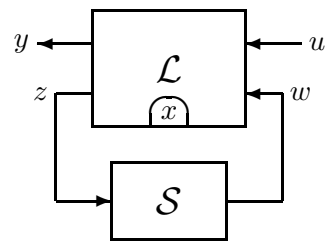


Figure 2.3: Canonical model structure

which in the interconnection become inputs to $\mathcal{L}$, and y are the output signals which in the interconnection are among the outputs of $\mathcal{L}$.

This looks like a "linear fractional transformation", a tool which is commonly used in the controls field to identify and treat separately specific types of model components. Loosely speaking, the LFT represents the system of equations

$$(y, z) = \mathcal{L}(x_1, u, w) \tag{2.4a}$$

$$w = \mathcal{S}(z). \tag{2.4b}$$

If the following is a state-space representation of $\mathcal{L}$,

$$\mathcal{L} = \left[\begin{array}{c|cc} A(k) & B_u(k) & B_w(k) \\ \hline C_y(k) & D_{yu}(k) & D_{yw}(k) \\ C_z(k) & D_{zu}(k) & D_{zw}(k), \end{array}\right]$$

then this is a state-space form of the interconnection:

$$x(t+1) = A(t)x(t) + B_u(t)u(t) + B_w(t)w(t) \tag{2.5a}$$

$$y(t) = C_y(t)x(t) + D_{yu}(t)u(t) + D_{yw}(t)w(t) \tag{2.5b}$$

$$z(t) = C_z(t)x(t) + D_{zu}(t)u(t) + D_{zw}(t)w(t) \tag{2.5c}$$

$$w(t) = \mathcal{S}(z(t)). \tag{2.5d}$$

It's possible to eliminate z and w from these equations; combining the last two for time t,

$$z = C_z x + D_{zu}u + D_{zw}S(z),$$

and solving for z

$$z = (I - D_{zw}S)^{-1}[C_z x + D_{zu}u].$$

where I is the identity operator and we are assuming the inverse of the operator $I(\cdot) - D_{zw}S(\cdot)$ is well-defined, a condition that's generally satisfied for LFTs that result from rearranging a set of nonlinear difference equations. Then w is given by

$$w = S(z) = S((I - D_{zw}S)^{-1}[C_z x + D_{zu}u]),$$

resulting in the reduced system equations

$$x(t+1) = A(t)x(t) + B_u(t)u(t) + B_w(t)(S \circ (I - D_{zw}S)^{-1})[C_z(t)x(t) + D_{zu}(t)u(t)] \tag{2.6a}$$

$$y(t) = C_y(t)x(t) + D_{yu}(t)u(t) + D_{yw}(t)(S \circ (I - D_{zw}S)^{-1})[C_z(t)x(t) + D_{zu}(t)u(t)]. \tag{2.6b}$$

For a given $\mathcal{L}$ and $\mathcal{S}$, we denote the system defined in Figure 2.3 as $(\mathcal{L} \circledast \mathcal{S})$. This, along with Figure 2.3, Equations (2.4), Equations (2.5), and Equations (2.6) are several ways to represent the same interconnection of $\mathcal{L}$ and $\mathcal{S}$.

The expression of models in the LFT form is useful because it lets us treat model sets having a wide diversity of structures under a common framework. Conversely it allows us to devise system identification methods which can be applied to a large class of problems with their own unique types of structure, once the problem has been cast as an LFT. This is because it separates model components into linear and static which can then be treated differently, taking advantage of the properties of linearity and staticness respectively.

The first question of interest is, how general a class of models can be represented as an interconnection of a linear and a static part? The short answer is that in fact *any* discrete-time state-space system can be written like this. Because, we can rewrite the general system equations (2.1) in the following way.

$$\begin{bmatrix} x(t+1) \\ y(t) \\ z_1(t) \\ z_2(t) \\ z_3(t) \end{bmatrix} = \underbrace{\left[\begin{array}{c|cccc} 0 & 0 & 0 & I_{n_x} & 0 \\ \hline 0 & 0 & 0 & 0 & I_{n_y} \\ I_{n_x} & 0 & 0 & 0 & 0 \\ 0 & I_{n_u} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{array}\right]}_{\mathcal{L}} \begin{bmatrix} x(t) \\ u(t) \\ t \\ w_1(t) \\ w_2(t) \end{bmatrix}, \tag{2.7a}$$

and

$$\begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix} = \underbrace{\begin{bmatrix} f(z_1(t), z_2(t), z_3(t)) \\ h(z_1(t), z_2(t), z_3(t)) \end{bmatrix}}_{\mathcal{S}}. \tag{2.7b}$$

The $\mathcal{L}$ and $\mathcal{S}$ pointed out have the required features: $\mathcal{L}$ is linear (dynamic, and indeed time-invariant), and $\mathcal{S}$ is static. Notice that in this representation, all of the information that defines the system is contained in $\mathcal{S}$. (Here we need to make time an explicit input signal, in order that the lower block is in fact a static function of its inputs. Since time is presumably known, this presents no loss of generality. When f and h do not depend explicitly on time, and even sometimes when they do (by using an LTV $\mathcal{L}$), it is not necessary to consider time as an input.)

Next consider the mass-spring system in Example 2.1, and how it might be rearranged into an LFT of a linear and a static system. Consider the system equations (2.2) and (2.3) for this model, for some choice of $T, m_1, m_2, c_1, k_1(\cdot), k_2(\cdot)$, and $f_2(\cdot)$. As we just saw, one way to do it is to define $\mathcal{L}$ and $\mathcal{S}$ as in (2.7a) and (2.7b), where it is clear from the system equations what are the appropriate f and h to use.

However there are other ways to put the mass-spring model into the LFT form. A natural choice is to pull out only the individual nonlinear components of the dynamic system into $\mathcal{S}$. To do this, in the system equations (2.2) and(2.3) we look to replace any nonlinear parts by signals defined as static (nonlinear) functions composed with linear functions of the state and inputs. For instance the signal $k_1(x_1 - x_2)$ is a static function of $x_1 - x_2$. $\mathcal{L}$ will then be defined so that the linear combinations of states/inputs are outputs, and $\mathcal{S}$ is composed of the static functions. The result is something like

$$\begin{bmatrix} x(t+1) \\ y(t) \\ z_1(t) \\ z_2(t) \\ z_3(t) \end{bmatrix} = \underbrace{\left[\begin{array}{cccc|ccccc} 1 & 0 & T & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & T & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-\frac{T}{m_1} & 0 & \frac{T}{m_1} & 0 & -\frac{T}{m_1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \frac{T}{m_2} & -\frac{T}{m_2} & -\frac{T}{m_2} \\ \hline 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right]}_{\mathcal{L}} \begin{bmatrix} x(t) \\ u(t) \\ e(t) \\ w_1(t) \\ w_2(t) \\ w_3(t) \end{bmatrix}, \tag{2.8a}$$

and

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \underbrace{\begin{bmatrix} k_1(z_1) \\ k_2(z_2) \\ f_2(z_3) \end{bmatrix}}_{\mathcal{S}(z_1,z_2,z_3)} \tag{2.8b}$$

This representation is in some sense more compact than the LFT for the general state-space system. There are fewer signals interconnecting the linear and static parts. More of the information is contained in $\mathcal{L}$, and $\mathcal{S}$, made up of the component nonlinearities of the model equations, is smaller in the sense that its these components have fewer inputs. The input/output dimensions of $\mathcal{L}$ and $\mathcal{S}$ has a direct effect on the size of the algorithms which are to come, and on a more fundamental level determines the dimension of the space we're searching over, that is the dimension of the model set. This is an important consideration.

Recall in Example 2.1 that the system identification goal was to identify suitable functions $k_1(\cdot), k_2(\cdot)$, and $f_2(\cdot)$. In this latter LFT representation of the models considered there, these three unknown functions appear in the static block $\mathcal{S}$. It is precisely to treat this type of situation that we are interested in developing new methods for system identification.

2.4 Problem Statement

Now we are ready to describe the types of system identification questions that are the focus of much of this dissertation, and give a general formulation for the problems we consider. In short, we attack the problem of identifying unknown static nonlinear components in a model.

We seek to identify discrete-time nonlinear state-space models. The measured inputs and outputs of the data-generating system are represented by u and y in the model, respectively. Together these represent the empirical input-output data available for the system identification task. Additionally a model may contain an unmeasured stochastic disturbance input e which can account for inexact and/or incomplete measurements, and mismatch between the model set and the physical system.

Further, we work with models that are in the LFT form described in Section 2.3. With the additional input e shown explicitly, the block diagram for such a model is shown in Figure 2.4.

We consider the situation where the linear part is known and the static part is to be identified. To be more exact, we consider model sets that are generated by the interconnection of some fixed linear system with all static operators that are consistent with a desired structure. We might express this as

$$\mathfrak{M} = \{(\mathcal{L} \circledast \mathcal{S}) : \mathcal{S} \text{ has the required structure}\}.$$

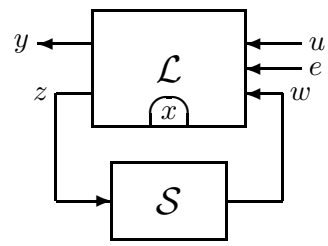


Figure 2.4: Canonical model set

The task here is to find the $\hat{\mathcal{S}}$ from this set which is optimal in terms of the identification criterion.

In doing this the interconnection signals z and w may or may not be available, that is, they may or may not be part of the output or input vectors y and u. Of course to the extent the input and output signals of $\mathcal{S}$ are known this makes the task of identifying $\mathcal{S}$ easier, but we try not to assume anything here.

Staticness is one property we obviously require the identified $\mathcal{S}$ to have. Other constraints arise from *a priori* information or desired characteristics of the identified model. These can reflect physical insight into a particular process, or experiential knowledge about the elements needed for a model to be a good representation of some process or system. In the common case of working from a set of state-space equations obtained from modeling and rearranging this into the LFT form, structure and knowledge contained in the former is reflected in the latter.

Four specific kinds of structure that are commonly seen are the following.

1. $\mathcal{S}$ is *block-diagonal*, by which we mean that certain outputs do not depend on certain inputs. For instance a two-input, two-output $\mathcal{S}$ may be composed of two single-input, single-output functions:
$$\mathcal{S}(z_1, z_2) = \begin{bmatrix} \mathcal{S}_1(z_1) \\ \mathcal{S}_2(z_2) \end{bmatrix}.$$
Here $\mathcal{S}_1$ does not depend on z_2, and $\mathcal{S}_2$ does not depend on z_1.
2. Some parts of $\mathcal{S}$ are known, or partially known, or known to within some uncertainty bounds.
3. Some elements of $\mathcal{S}$, although unknown, may be known to be equal to, or related in certain other ways to, other elements of $\mathcal{S}$. For example a given nonlinearity may be

repeated twice. Or, one function might be the derivative of another.

4. $\mathcal{S}$ may be required to have certain smoothness properties, such as Lipschitz bounds, bounds on magnitudes of derivatives, and such.

This is the main problem we attack in this dissertation. That of identifying the static components of these types of model sets, using the available input-output information, and respecting any desired structure. Our methods are nonparametric in nature. We will focus on handling some of these various types of structural constraints; some are harder than others.

For the mass-spring example and the system ID problem described at the end of Example 2.1 in which the three nonlinearities $k_1(\cdot), k_2(\cdot)$, and $f_2(\cdot)$ are unknown, we saw an LFT form of that model, (2.8), in which the parameters of the linear system are considered to be known and the elements of $\mathcal{S}$ are the ones to be identified. Here $\mathcal{S}$ has a block-diagonal structure; $\mathcal{S}_1$ depends only on the first input to $\mathcal{S}$, $\mathcal{S}_2$ only on the second, and $\mathcal{S}_3$ only on the third. In the case that $k_1 = k_2$, then there also is a nonlinearity which is repeated. Thus this example fits naturally into the problem formulation with this choice of model set.

In a sense the problem formulation is very general. In the LFT (2.7) for the general discrete-time system, the linear part is known, and the state transition and readout functions which determine the system are located in $\mathcal{S}$. So for any discrete-time system, there is a model set in this formulation of linear part and static part, with the linear part known, which contains that system, and using this model set we can hope to identify it.

For instance we could also formulate the identification problem for the mass-spring example in this way. Then we would be identifying static functions f and h such that the LFT (2.7) is a good model. In doing this we lose the structure and/or apriori information we gained from modeling, and the size of the static functions we need to identify (their input and output dimensions) grows. In the compact representation (2.8), the model set has three single-input nonlinearities to identify. But in (2.7) the model set has two nonlinearities with six inputs each (we don't need to use t as an input for this example). In general it is a vastly more challenging task to identify, or even represent in a computer, a function whose domain is six-dimensional, than one with one input. And indeed for our methods, the smaller the input and output dimensions of the static functions we are identifying, the better they work, and the more computationally tractable they are.

The bottom line is, the choice of LFT representation for a given identification

problem is nonunique. For a given structured model set, in general there are many structured LFT model sets which *contain* it. The choice of which one to use has important implications for how well things are going to work out. If we choose one that is larger than the "natural" model set then we may lose structure and/or information, and in general the size of the identification problem grows. On the other hand, the choice is constrained by available methods, which may not apply to all structured LFT model sets. These issues have important implications for the generality and limitations of our methods.

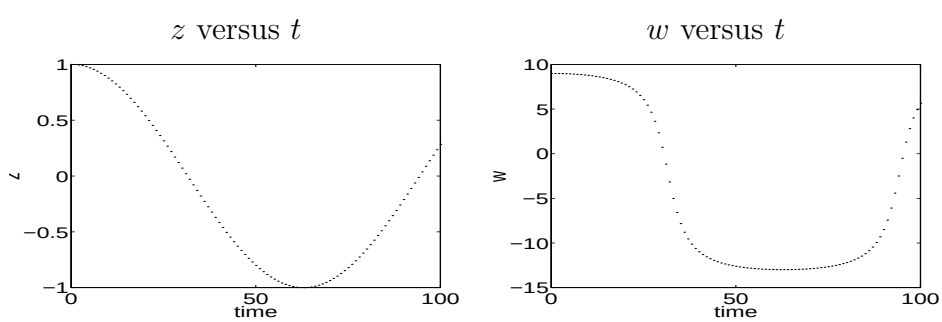
Chapter 3

Elements

3.1 Approach

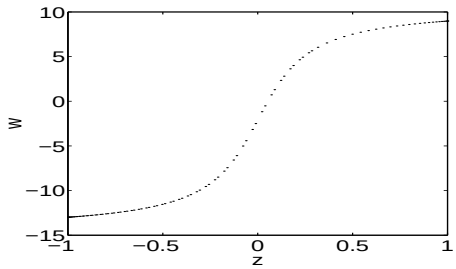
Working toward solving the problem outlined in Section 2.4, in which the goal is to identify the unknown static mappings of $\mathcal{S}$, suppose we conduct an experiment of length L during which we record $u_{[1,L]}$ and $y_{[1,L]}$. Assuming that the data-generating system has the structure of Figure 2.3, it makes sense to talk about the signals z and w that were, respectively, the inputs and outputs of the operator $\mathcal{S}$ during the experiment. If we somehow had access to $z_{[1,L]}$ and $w_{[1,L]}$ then this of course provides information about $\mathcal{S}$. More specifically for each t, the pair $(z(t), w(t))$ is an element of the graph of the function. Recall the graph of a function $f : D \to R$ is the set $\{(x, f(x)) : x \in D\}$. In a sense the set of points$\{(z(t), w(t))\}_{t=1}^{L}$ is the most basic information about $\mathcal{S}$ contained in the experimental data. This set is where the function was sampled during the experiment.

For example consider the following two signals which are a sequence of inputs $(z(t))$ and the corresponding sequence of outputs $(w(t)) = (\mathcal{S}(z(t)))$ for a particular SISO operator.



For these signals, the scatter plot of the input-output pairs looks like

scatter plot: $\{(w(t), z(t))\}$



Most of us will look at this set of points and feel confident we have a good understanding of $\mathcal{S}$, at least over the interval $[-1, 1]$.

In reality, since we usually work with functions defined on uncountable domains, this partial graph, being a finite number of points, is a rather small subset of the entire graph. To infer something about the rest of it involves assumptions about the properties and intersample behavior of $\mathcal{S}$, e.g. smoothness properties. The points can be further interpolated or fitted to form an estimate of the entire function for instance by linear interpolation, construction of a spline curve, or a best polynomial fit. Smoothness assumptions can then allow us to bound the mismatch between the estimate and the actual function. Intuitively speaking, in this example it seems pretty clear what is going on without the need to explicitly think about a procedure for extending the graph between or beyond the samples, due to our native preference for smooth functions.

In any case the problem reduces to two successive steps: the first of determining this partial graph of the function and forming the scatter plot, and the second of using these points to form an estimate of the relation or function contained therein. The second step of fitting a function to a set of points is a common engineering task. However the task of estimating the scatter plot is more novel, and this is what we address.

The challenge is that we do not in general get to choose which signals are accessible or easy to measure, and since the static functions we're interested in are subcomponents of the larger system whose inputs and outputs we measure, their particular inputs and/or outputs may not be available. Since we would like an identification method that doesn't depend on these signals being known or measured, for any unknown or unmeasured parts of $z_{[1,L]}$ and $w_{[1,L]}$, we *estimate* the values of those signals that were realized during the experiment. In doing so we form an estimate for a portion of $\mathcal{S}$'s graph. The criteria for choosing those estimates is based on the available input-output data from the larger system,

along with other knowledge and intuition about the system and unknown signals involved in the problem. The goal of course is to achieve on average the best estimates possible and therefore the most accurate picture of $\mathcal{S}$.

Looking at the canonical candidate model set of Figure 2.4, we consider the situation that part or all of the signals e, w, z and initial condition x_1 may be unknown for a given experiment. The probability of different realizations of e and x_1 affect what z and w are likely to have been, and so the approach is to estimate all of the unknown portions of these signals.

Perhaps the most evident information we have is the input-output data. The linear system $\mathcal{L}$ defines a relationship between the signals $u,\ e,\ w,\ y,\ z$, and the system's initial condition. If $\mathcal{L}$ is assumed to be known then we have a set of linear equality constraints among these signals. Some of the elements are known or measured. In estimating the remaining unknown elements we will require that they are consistent with the linear system and experiment data. This begins to narrow down the choices for unknown inputs and outputs of $\mathcal{S}$. A more detailed discussion of this constraint and efficient ways to handle it is given in Section 3.2.

Another source of constraints and criteria comes from assumptions about $\mathcal{S}$ and properties that we require an identified representation of it to possess. These criteria guide the estimates of the partial graph of $\mathcal{S}$. Foremost among these properties is that $\mathcal{S}$ is a static operator with some reasonable smoothness; the estimates for (z, w) should be congruous with these and there are various ways to enforce this. If $\mathcal{S}$ has input-output structure it is important to take that into account. It is also possible to take advantage of knowledge of elements that are repeated or related or have known smoothness properties. Taken together, these considerations can be applied in various ways to rule out or penalize various combinations of signal estimates. The issue of analyzing if signals are compatible with a static operator, as well as some of these other constraints on $\mathcal{S}$, is the focus of Section 3.3.

A third important source of constraints on and/or fitness criteria for the estimates are known or assumed statistical properties of the disturbance signal e such as its mean and variance, autocovariance and spectrum, and correlation with other signals (or lack thereof). Most fundamentally, we would like any estimate for e to be a likely sample path according to the assumed stochastic model. It should be consistent with probabilistic assumptions. By limiting the realizations of the noise that will be considered or preferring some to others, this limits the estimates for z and w to be considered or makes some more preferred than

others, via other constraints and criteria that relate e to z and w. This is covered in Section 3.4.

These are the three main criteria that we consider, but we don't exclude other possibilities that might improve the estimates, especially if they are easy to incorporate. We will be estimating directly some of the signals involved in the block diagram of Figure 2.3, that is the decision variables in our optimization problems are essentially these sequences of numbers. To the extent that a particular criteria can be expressed directly in terms of these signals, the more naturally it will fit into such an optimization problem. Some examples of easily-incorporated constraints are bounds on maximum magnitude of a signal, or total energy content.

To summarize, the goal is to search for "good" estimates of the signals we're interested in, namely the unknown inputs and outputs of $\mathcal{S}$. In view of the diverse types of *a priori* information and model structure specifications, there is a wide variety of constraints and criteria that might be used to guide us in searching for good estimates. It is natural to formulate various optimization problems to capture these criteria and locate the best estimates. A generic form of the type of optimization problem we expect to formulate is

$$\begin{array}{ll} \underset{x \in \mathbf{R}^n}{\text{minimize}} & V(x) \\ \text{subject to} & f_i(x) \leq 0 \quad i = 1..p \\ & g_i(x) = 0 \quad i = 1..r \end{array} \tag{3.1}$$

The decision variables x include unknown parts of z and w as well as other unknowns and degrees of freedom in the problem, such as the noise signal e and initial conditions x_1. The $f_i(\cdot)$ and $g_i(\cdot)$ represent inequality and equality constraints respectively and $V(x)$ is an objective function which judges the quality of a candidate estimate. The constraints and objective are derived from the basic criteria just described:

1. the linear system constraints are satisfied
2. z and w are compatible with a static operator which has the correct structure and satisfies other apriori assumptions
3. e is consistent with stochastic assumptions
4. other.

To formulate an optimization problem is one thing but to actually be able to solve it is another. Ideally we could assemble optimizations which accurately reflect the problems we want to solve, which use the constraints and criteria we think are most natural. But to

get solutions we have to work within the constraint of what is currently possible, in terms of computational power and solution methods. These types of questions are especially important in light of the large parameter space being searched—the number of scalar decision variables will be roughly an integer multiple of the data length, which is often necessarily large. We often need to compromise and settle for criteria that lend themselves to computational tractability. Therefore we look to express the various criteria and constraints discussed above in ways that enable us to apply specific favorable optimization frameworks and algorithms, and we look to simplify the problem whenever possible.

Linear programming (LP) and convex quadratic methods (such as least squares) are classes of optimization problems with well-known solution algorithms that work well in practice with large decision vectors. Semidefinite programming (SDP; also known as linear matrix inequality problems) handles more general problem formulations and includes LP and least squares. Relatively large problem sizes are also possible here. SDP algorithm development is an active area of research, especially in terms of exploiting the structure of particular semidefinite problems [41]. These classes of problems are considered, along with the more general class of convex programming (CP), and sometimes in combination with more ad-hoc procedures. A main question is, given a particular framework such as one of these, is it possible to include these criteria/constraints in that framework. Much of the work in this dissertation is focused on these issues.

As discussed in Chapter 1, although the distinction between parametric and nonparametric methods is somewhat fuzzy, the approach we are suggesting is most naturally classified as nonparametric. True, we estimate a finite-length discrete time signal, which is a finite set of parameters. However the underlying goal is to estimate a function. Also, the number of parameters is dependent on the length of the data set and is not a defined property of the model set. As the experiment grows, the number of parameters used to estimate $\mathcal{S}$ grows, which allows that the estimate becomes more and more descriptive. Lastly, the parameters we estimate do not have an interpretation as some intrinsic property of the data-generating system, rather they are a property of the experiment.

The next three sections cover the three main elements of our estimation procedure in more detail, namely linear system constraints, staticness criteria, and stochastic criteria. We propose and discuss several quantitative measures that capture various notions of the fitness of signal estimates, keeping in mind that we intend to later use versions of these measures within specific optimization frameworks. With respect to this each has strong

and weak points which will be explored in later analysis and examples.

3.2 Linear System Constraints

As described in the previous section, our approach is to perform experiments and then estimate the unknown signals of interest, in an optimization setting. We only consider estimates that are consistent with the known linear system $\mathcal{L}$ and experimental data. This can be taken care of by adding constraints equations to the optimization formulation, and choosing a solver that can perform the constrained optimization.

A more attractive option is to compute an explicit parametrization of this constraint set's feasible set and reformulate the problem in terms of the free variables of the parametrization, rather than describing the set implicitly with constraint equations. This can work especially well with linear constraints in the setting of convex programming. It eliminates the constraint equations and reduces the number of decision variables, making for an easier optimization. In the present situation this is important, due to the large number of constraint equations and variables involved.

Since we're working in a discrete-time setting over a finite time interval, the constraints involved take the form of a finite number of linear equations among a finite number of unknowns. This can be written as a matrix equation, and the parametrization of the solutions to this equation can be computed using standard matrix operations. This approach is described in the next section. However for long enough data records this becomes infeasible in practice, due to difficulties associated with unstable operators and sheer problem size. Later sections develop alternative methods to represent and compute the parametrization which take advantage of the special dynamic-linear-system structure of these constraints in order to derive computations which are more efficient and overcome these problems.

3.2.1 Parametrization of the Feasible Set of the Linear System Constraint

The linear system constraints between the various signals defined in the model structure of Figure 2.3 is shown pictorially in Figure 3.1 (which is just Figure 2.3, minus the static function block). Let the following be a state space representation of $\mathcal{L}$

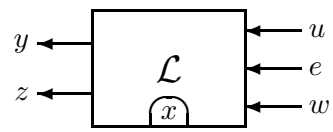


Figure 3.1: Linear system constraints

$$x(n+1) = \ Ax(n) + \ B_u u(n) + \ B_e e(n) + B_w w(n) \tag{3.2a}$$

$$y(n) = C_y x(n) + D_{yu} u(n) + D_{ye} e(n) + D_{yw} w(n) \tag{3.2b}$$

$$z(n) = C_z x(n) + D_{zu} u(n) + D_{ze} e(n) + D_{zw} w(n), \tag{3.2c}$$

These equations, with n ranging from 1 to L, define the output vectors $y_{[1,L]} \in \mathbf{R}^{n_y L}$ and $z_{[1,L]} \in \mathbf{R}^{n_z L}$ as a linear function of the vector $(x_1, u_{[1,L]}, e_{[1,L]}, w_{[1,L]}) \in \mathbf{R}^{n_x} \times \mathbf{R}^{n_u L} \times \mathbf{R}^{n_e L} \times \mathbf{R}^{n_w L}$, where x_1 is the state at time $n = 1$. Since we're working in a discrete-time setting over a finite time interval, this dependence can be written out in terms of matrices as

$$y_{[1,L]} = T_{yx_1} x_1 + T_{yu} u_{[1,L]} + T_{ye} e_{[1,L]} + T_{yw} w_{[1,L]} \tag{3.3a}$$

$$z_{[1,L]} = T_{zx_1} x_1 + T_{zu} u_{[1,L]} + T_{ze} e_{[1,L]} + T_{zw} w_{[1,L]}. \tag{3.3b}$$

Here e.g. T_{yu} is the block Toeplitz matrix

$$T_{yu} := \begin{bmatrix} D_{yu} & 0 & 0 & \cdots & 0 \\ C_y B_u & D_{yu} & 0 & \cdots & 0 \\ C_y A B_u & C_y B_u & D_{yu} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_y A^{L-2} B_u & C_y A^{L-3} B_u & C_y A^{L-4} B_u & \cdots & D_{yu} \end{bmatrix} \tag{3.4}$$

which is the matrix representation of the the zero-state response of $\mathcal{L}_y$ to u. Similarly $T_{ye}, T_{yw}, T_{zu}, T_{ze}$, and T_{zw} are zero-state responses, while T_{yx_1} and T_{zx_1} bookkeep the response of the system to initial conditions, for instance

$$T_{yx_1} := \begin{bmatrix} C_y \\ C_y A \\ C_y A^2 \\ \vdots \\ C_y A^{L-1} \end{bmatrix}. \tag{3.5}$$

We could just as easily handle an LTV $\mathcal{L}$ by defining the T_* appropriately.

As discussed in Section 3.1, we require that any estimates for e, w, z, and x_1 are consistent with $\mathcal{L}$ and the input-output data. That is, they satisfy equations (3.3a) and (3.3b). Estimates that do so are called *feasible* for the linear system constraints. As the solution set of a system of linear equations the feasible set is an affine subspace of the parameter space $\mathbf{R}^{n_x} \times \mathbf{R}^{n_e L} \times \mathbf{R}^{n_w L} \times \mathbf{R}^{n_z L}$. The optimization (3.1) can be thought of as checking each point in this set

$$\left\{(x_1, e_{[1,L]}, w_{[1,L]}, z_{[1,L]}) \subseteq \mathbf{R}^{n_x} \times \mathbf{R}^{n_e L} \times \mathbf{R}^{n_w L} \times \mathbf{R}^{n_z L} : \text{(3.3a) and (3.3b) hold}\right\}$$

for how well it fits the rest of our prior knowledge and assumptions, and choosing the one that is most consistent.

We assume that for a given data record, the system of equations (3.2a) and (3.2b) is consistent, which is to say the system is underdetermined, and there is at least one choice for the unknown quantities that satisfies the linear system constraints. In this case we look to other criteria (staticness, stochastics, etc) to select a good one.

One situation where a solution is guaranteed to exist regardless of the input-output data is that each of the n_y measurements is noisy and this is modeled as an unknown output error added to each measurement. In this case for any input-output data record (u, y) there are many solutions (x_1, e, w, z) to the linear system. This is because D_{ye} is full rank, and so $D_{ye}D_{ye}^T$ is invertible, which in turn means $T_{ye}T_{ye}^T$ is invertible. Then for any choice of $w_{[1,L]}$ and x_1 we can solve (3.3a) for $e_{[1,L]}$; one possibility is the least-squares solution for an underdetermined system of equations:

$$e_{[1,L]} = T_{ye}^T(T_{ye}T_{ye}^T)^{-1}(y_{[1,L]} - T_{yu}u_{[1,L]} - T_{yw}w_{[1,L]} - T_{yx_1}x_1)$$

Then choose $z_{[1,L]}$ according to (3.3b), and this choice of $(x_1, e_{[1,L]}, w_{[1,L]}, z_{[1,L]})$ satisfies the linear system. In effect, we can attribute any discrepancy in the output y to noise if necessary.

It would be straightforward at this point, given $\mathcal{L}$ and input-output data, to write out the constraint equations (3.2a) and (3.2b) and pass them, along with additional criteria (as alluded to in Section 3.1) to a routine which computes the optimal estimates. However due to the nature of the problem this is typically a very large number of equations, and a very large number of free variables. The optimization quickly becomes very challenging

when dealing with systems with multiple inputs and outputs, many unknowns, and long data records.

A common practice is to eliminate some or all of the equality constraints before passing the optimization problem on to a solver. A little preprocessing beforehand can result in an formulation which is smaller and more easily solved, the solution is equivalent to a solution of the original problem [41]. Looking at the generic problem (3.1), suppose that we can find a parametrization of the solution set of the equality constraints. That is we find a function $\chi : \mathbf{R}^f \to \mathbf{R}^n$ such that

$$\{x \in \mathbf{R}^n : g_i(x) = 0, \quad i = 1..r\} = \{K(f) : f \in \mathbf{R}^\kappa\}.$$

Then solving (3.1) is equivalent to solving

$$\begin{array}{ll} \underset{f \in \mathbf{R}^\kappa}{\text{minimize}} & V(K(f)) \\ \text{subject to} & f_i(K(f)) \leq 0 \quad i = 1..p \end{array}, \tag{3.6}$$

in the sense that if f is optimal for (3.6) then $x = K(f)$ is optimal for (3.1), and if x is optimal for (3.1) then there exists an f, optimal for (3.6), with $x = K(f)$. The hope is that, with elimination of equality constraints and a possible reduction in the number of free variables, this is a easier problem to solve than the original.

This can work especially well when the constraints are linear. In general one can parametrize the solution set of a system of linear equations as a particular solution, plus anything in the nullspace of the linear operator. To be concrete suppose the equality constraints are represented by a system of linear equations

$$Ax = b,$$

with $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$. Suppose that this constraint is feasible and there is at least one solution, i.e. $b \in \mathcal{R}(A)$. Let x^0 be a particular solution, and $K \in \mathbf{R}^{n \times p}$ such that $\mathcal{R}(K) = \mathcal{N}(A)$. Then

$$\{x \in \mathbf{R}^n : Ax = b\} = \{x^0 + Kf : f \in \mathbf{R}^p\}. \tag{3.7}$$

The rank of K is necessarily equal to $\kappa := \dim \mathcal{N}(A) = n - \text{rank}(A)$, the nullity of A. Thus if $K \in \mathbf{R}^{n \times \kappa}$ then K has a trivial nullspace, and for each solution x there is exactly one vector $f \in \mathbf{R}^\kappa$ such that $x = x^0 + Kf$, and so this is a sort of minimal parametrization of the nullspace of A.

In this case the original problem is transformed to

$$\begin{array}{ll} \underset{f \in \mathbf{R}^{\kappa}}{\text{minimize}} & V(x^0 + Kf) \\ \text{subject to} & f_i(x^0 + Kf) \leq 0 \quad i = 1..p \end{array}, \tag{3.8}$$

Here the equality constraints are eliminated. Typically the set of decision variables that are consistent with the equality constraints are a lower-dimensional manifold of the full decision variable space, i.e. $\kappa < n$; in fact $\kappa - n = \text{rank}(A)$.

One potential drawback to doing this is that the structure of the transformed cost and constraint functions may be less convenient for the optimization procedure to handle. In some cases it can be more efficient to leave the problem in its original form, even with a greater number of constraints and decision variables, because the objective and constraint functions have an advantageous structure which the optimization procedure can take advantage of. For our purposes it is important that if $g(x) : \mathbf{R}^n \to \mathbf{R}^p$ is convex in x, then $h(f) = g(x^0 + Kf)$ is convex in f. That is, the composition of a convex with an affine function is still convex. We can replace "convex" with "affine" or "quadratic" and the statement is still true. Also, LMIs remain LMIs. Since the unknown signals and initial condition have been parametrized as affine functions of the new decision variable, any problem that was convex (linear, quadratic, LMI) in x is still convex (linear, quadratic, LMI) in f after elimination of the equality constraints.

Apply these ideas to the linear system constraints. First note that for any choice of e, w, and x_1 which satisfies (3.3a), there is always a choice of z such that (3.3b) is also true (dropping the $_{[1,L]}$ subscript for now). Thus we should consider the set of e, w, and x_1 such that the y equation holds, and parametrize the unknown z accordingly. Rewrite the y equation as

$$\begin{bmatrix} T_{yx_1} & T_{ye} & T_{yw} \end{bmatrix} \begin{pmatrix} x_1 \\ e \\ w \end{pmatrix} = y - T_{yu}u. \tag{3.9}$$

Here the right-hand side is a known quantity. Let (x_1^0, e^0, w^0) be a particular solution of this, κ the dimension of the nullspace of $\begin{bmatrix} T_{yx_1} & T_{ye} & T_{yw} \end{bmatrix}$, and $K \in \mathbf{R}^{(n_x + n_e L + n_w L) \times \kappa}$ a matrix whose range space is equal to the nullspace of this matrix, i.e. K has independent

columns and $\begin{bmatrix} T_{yx_1} & T_{ye} & T_{yw} \end{bmatrix} K = 0$. Partition K conformably as

$$K = \begin{bmatrix} K_x \\ K_e \\ K_w \end{bmatrix}.$$

Apparently we can parametrize the set of (x_1, e, w) which are consistent with the linear system and input-output data as

$$\left\{ \begin{pmatrix} x_1 \\ e \\ w \end{pmatrix} : \text{(3.3a) holds} \right\} = \left\{ \begin{pmatrix} x_1^0 + K_x f \\ e^0 + K_e f \\ w^0 + K_w f \end{pmatrix} : f \in \mathbf{R}^\kappa \right\}. \tag{3.10}$$

The signal z^f that is consistent with the linear system, input-output data, and particular choice of f is

$$\begin{aligned} z^f &= T_{zx_1} x_1^f + T_{zu} u + T_{ze} e^f + T_{zw} w^f \\ &= T_{zx_1}(x_1^0 + K_x f) + T_{zu} u + T_{ze}(e^0 + K_e f) + T_{zw}(w^0 + K_w f) \\ &= \underbrace{(T_{zx_1} x_1^0 + T_{zu} u + T_{ze} e^0 + T_{zw} w^0)}_{z^0} + \underbrace{\begin{bmatrix} T_{zx_1} K_x & T_{ze} K_e & T_{zw} K_w \end{bmatrix}}_{K_z} f \\ &= z^0 + K_z f, \end{aligned} \tag{3.11}$$

defining z^0 and K_z as indicated.

In the sequel f denotes the decision variables. The unknown signals and/or initial condition are parametrized in terms of f as

$$x_1^f = x_1^0 + K_x f \tag{3.12a}$$

$$e^f = e^0 + K_e f \tag{3.12b}$$

$$w^f = w^0 + K_w f \tag{3.12c}$$

$$z^f = z^0 + K_z f \tag{3.12d}$$

The linear equality constraints are "built-in" to this parametrization. The size of the decision vector f is in general smaller than, and never larger than, the original decision vector involving the unknown parts of (x_1, e, w, z).

For the solution parametrization it's necessary to compute a particular solution of a certain system of equations, as well as a basis for the nullspace of a particular linear

operator. When working with matrix representations of the operators in question, solutions for these problems are readily available.

In general there are many solutions of (3.9) for the unknown parts of (x_1, e, w). One choice is the minimum-norm solution, which is an optimal point of the minimization

$$\begin{array}{ll} \text{minimize} & \|(x_1, e, w)\|_2 \\ \text{subject to} & \begin{bmatrix} T_{yx_1} & T_{ye} & T_{yw} \end{bmatrix} \begin{pmatrix} x_1 \\ e \\ w \end{pmatrix} = y - T_{yu}u. \end{array} \tag{3.13}$$

This is a least-squares problem, and solvers with good numerical properties are available.

One way to characterize the nullspace of a matrix and find basis vectors which span it is to compute its singular value decomposition (SVD). For background on the SVD see e.g. [30]. Suppose $A \in \mathbf{R}^{m \times n}$ with singular value decomposition

$$\begin{aligned} A &= USV^* \\ &= U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^* \\ V_2^* \end{bmatrix} \end{aligned}$$

such that $U \in \mathbf{C}^{m \times m}$, $S \in \mathbf{R}^{m \times n}$, and $V \in \mathbf{C}^{n \times n}$ with U and V unitary. Let r be the number of nonzero singular values, which is equal to the rank of A, and Σ is the $r \times r$ diagonal matrix of positive singular values. Let v_k be the k'th column of V. Then Av_k is nonzero for $1 \leq k \leq r$ and zero for $r < k \leq n$, and thus $\mathcal{N}(A)$ has dimension $n - r$, and is spanned by the last $n - r$ columns of V. Letting V_1 be the first r columns of V and V_2 the remaining, then the (independent) columns of V_2 span the nullspace of A:

$$\mathrm{Ker}(A) = \mathrm{span}(V_2). \tag{3.14}$$

We've been using matrices to represent discrete-time linear systems over finite lengths of time in part because they offer a nice concrete way to visualize what's going on. It's straightforward to see how to modify the procedures to handle time-varying systems, multiple experiments, various types of initial and boundary conditions, measurements that come at irregular sampling intervals, and other linear dependences of the output of the system on unknown parameters in a way that doesn't quite fit into the linear dynamic system framework we've been discussing. The parametrization of the feasible set of the linear equality constraints involves computing a particular solution, which may be done

via a least-squares or QR decomposition, and a basis for the nullspace of that operator, which can be found by computing a singular value decomposition. The sizes of the matrices involved in these computations is linear in the number inputs and outputs, as well as the time span considered. With currently available computers and algorithms, manipulation of large matrices is a viable solution method for many systems and quite large data records.

But for computational and numerical reasons it is not always convenient or even possible to deal with matrices. For one, when the length L of the data record is large, and when dealing with systems with many inputs and outputs, the matrices can quickly become too large to store, or to be able to perform computations such as SVD or QR decompositions.

Another potential problem is that if $\mathcal{L}$ is an unstable system then the Markov parameters such as CA^kB in the matrix representations can grow quickly with L. This presents issues with storing large numbers on a machine with finite word length. As well, Toeplitz matrices for unstable systems are typically ill-conditioned, once again causing problems for matrix operations one may attempt to perform on them.

It's useful to be able to handle unstable $\mathcal{L}$, even when the system to be identified is stable. The canonical separation of the system into linear and static parts as in Figure 2.3 can be an artificial division which results in component systems with properties different from the aggregate system. In particular even if the LFT is stable, $\mathcal{L}$ might unstable, being stabilized by the feedback through $\mathcal{S}$.

Finally, a system with nonminimum phase could also potentially pose a problem in computing particular solutions. We're solving a sort of inversion problem here, since $(e, w) \sim \mathcal{L}^{-1}(y)$, and the inverse of a non-minimum phase system is unstable. Using the least-norm particular solution choice as described above regularizes the problem, and so usually this isn't a big issue.

A linear dynamic system L defines an operator which maps an input signal u along with initial condition to an output signal y. One thing lost when thinking of this operator as a big matrix equation is the efficient recursive computation of the mapping. The matrix multiplication involves computing the outputs $y(t)$, for each time t, as a linear combination of the values of the inputs $u(t), \quad t = 1..L$. Hence to compute the output $u(t), \quad t = 1..L$ requires $O(L^2)$ operations. On the other hand for a causal system with a state description, the output and state at time t depend only on the input and state at time $t-1$, and the output can be calculated recursively (over t) in only $O(L)$ operations.

Since in this case the operator in question is indeed derived from a linear dynamic system, attractive alternatives exist for computing a particular solution and a basis of the nullspace. It is possible to take advantage of the recursive structure of the dynamic system equations to derive computations which are efficient in terms of computation and storage. Care is needed so that computations are also well-behaved; in particular any recursive operations we use in the solution process should be stable. In the next sections we consider representations of and methods for computing the data x_1^0, K_x, etc, in Equations (3.12). We want to handle a general configuration of unknown quantities in x_1, e, w, and z.

The next sections discuss how to efficiently compute particular solutions and a basis for the nullspace of the linear equality constraints. It's not essential to have read these in order to understand later sections. The short version of the story is that there's an efficient way to compute the min-norm particular solutions (3.13), and it's easy to find stable linear dynamic system representations of the operators K_* in (3.12a)–(3.12d) that characterize the nullspace of the linear equality constraints. There are a few assumptions about $\mathcal{L}$ that are used; mostly they make the derivations easier, and could be relaxed with some additional work. When working directly with matrices these assumptions aren't needed.

3.2.2 Linear Dynamic Systems and Operators They Define

This section mostly introduces notation and makes explicit what we mean by certain ideas and expressions used in the following two sections.

Given two linear operators, $L_1 : \mathbf{R}^{n_1} \to \mathbf{R}^m$ and $L_2 : \mathbf{R}^{n_2} \to \mathbf{R}^m$, by the *horizontal concatenation* $[L_1 \ L_2]$ we mean the operator $L : \mathbf{R}^{n_1} \times \mathbf{R}^{n_2} \to \mathbf{R}^m$ with action

$$L(x_1, x_2) = L_1 x_1 + L_2 x_2,$$

and borrowing from matrix notation we express $L(x_1, x_2)$ as

$$\begin{bmatrix} L_1 & L_2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

Going the other way we can *partition* a linear operator $L : \mathbf{R}^{n_1} \times \mathbf{R}^{n_2} \to \mathbf{R}^m$ into two operators according to its first n_1 and last n_2 inputs. Define

$$L_1 : \mathbf{R}^{n_1} \to \mathbf{R}^m, \quad x_1 \mapsto L(x_1, 0)$$
$$L_2 : \mathbf{R}^{n_2} \to \mathbf{R}^m, \quad x_2 \mapsto L(0, x_2),$$

and so

$$L_1(x_1) + L_2(x_2) = L(x_1, x_2).$$

Finally, given two linear operators $L_1 : \mathbf{R}^n \to \mathbf{R}^{m_1}$ and $L_2 : \mathbf{R}^n \to \mathbf{R}^{m_2}$, by the *vertical concatenation* $\begin{bmatrix} L_1 \\ L_2 \end{bmatrix}$ we mean the operator $L : \mathbf{R}^n \to \mathbf{R}^{m_1} \times \mathbf{R}^{m_2}$ with action

$$L(x) = (L_1 x,\ L_2 x)\,.$$

Roughly speaking, a linear dynamic system L maps an input signal u into an output signal y, and we write $y = Lu$ to symbolize this. But to be precise there are a number of different operators that can be associated with L, i.e. a number of ways to map one vector of numbers into another. Mainly we are concerned with distinguishing between

- the zero-state, or forced, response which maps an input vector $u(t),\ t = 1..N$ to an output vector $y(t),\ t = 1..N$,
- the zero-input, or free, response which maps an initial condition x_1 to output vector $y(t),\ t = 1..N$, and
- the full response which maps an initial condition x_1 plus input vector to output vector $y(t),\ t = 1..N$.

Other possibilities exist, for example an operator associated with L that maps an initial condition plus input vector to a vector of outputs along with a vector of states $x(t),\ t = 1..N$, or maybe one that maps an input vector to a final state. Mainly the distinctions come about as a result of using different subsets of the possible things to consider as inputs and outputs of the system. Being clear about which specific operator is being discussed is important in the following two sections, when talking notions such as the "inverse of L" and "adjoint of L".

Since we're working in discrete-time and with finite time spans, all linear operators that we consider have matrix representations. Some notation for the matrix representation of these three particular operators is useful. This notation makes it precise which of the possible operators corresponding to a given linear dynamic by defining its matrix representation, while bookkeeping the state-space data of the dynamic system it's derived from.

Define

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]_{\text{z-s}} := \begin{bmatrix} D_1 & 0 & 0 & 0 & 0 \\ C_2B_1 & D_2 & 0 & 0 & 0 \\ C_3A_2B_1 & C_3B_2 & D_3 & 0 & 0 \\ C_4A_3A_2B_1 & C_4A_3B_2 & C_4B_3 & D_4 & 0 \\ & & & & \ddots & \vdots \\ C_LA_{L-1}..A_2B_1 & C_LA_{L-1}..A_3B_2 & C_LA_{L-1}..A_4B_3 & C_LA_{L-1}..A_5B_4 & \dots & D_L \end{bmatrix} \tag{3.15}$$

This is shorthand for the matrix representation of the forced response of a discrete-time LTV causal dynamic system L with state-space data (A_k, B_k, C_k, D_k), which maps an input vector to the zero-state response as

$$y_{[1,L]_{\text{z-s}}} = \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]_{\text{z-s}} u_{[1,L]}. \tag{3.16}$$

For LTI systems this becomes the usual familiar toeplitz matrix

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]_{\text{z-s}} = \begin{bmatrix} D & 0 & 0 & 0 \\ CB & D & 0 & 0 \\ CAB & CB & D & 0 \\ & & & \ddots & \vdots \\ CA^{L-2}B & CA^{L-3}B & CA^{L-4}B & \dots & D \end{bmatrix}. \tag{3.17}$$

At times we use this notation when we want to be clear about which operator derived from the dynamic system we're talking about, in this case the zero-state response. Next define

$$\left[\begin{array}{c} A_k \\ \hline C_k \end{array}\right] := \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]_{\text{z-i}} := \begin{bmatrix} C_1 \\ C_2A_1 \\ C_3A_2A_1 \\ C_4A_3A_2A_1 \\ \vdots \\ C_LA_{L-1}..A_1 \end{bmatrix}. \tag{3.18}$$

This is shorthand for the matrix representation of the zero-input response of L:

$$y_{[1,L]_{\text{z-i}}} = \left[\begin{array}{c} A_k \\ \hline C_k \end{array}\right] x_1. \tag{3.19}$$

Putting these together, the matrix representation of the full response is

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right] :=$$

$$\begin{bmatrix} C_1 & D_1 & 0 & 0 & 0 & 0 \\ C_2A_1 & C_2B_1 & D_2 & 0 & 0 & 0 \\ C_3A_2A_1 & C_3A_2B_1 & C_3B_2 & D_3 & 0 & 0 \\ C_4A_3A_2A_1 & C_4A_3A_2B_1 & C_4A_3B_2 & C_4B_3 & D_4 & 0 \\ & & & & & \ddots & \vdots \\ C_LA_{L-1}..A_1 & C_LA_{L-1}..A_2B_1 & C_LA_{L-1}..A_3B_2 & C_LA_{L-1}..A_4B_3 & C_LA_{L-1}..A_5B_4 & \dots & D_L \end{bmatrix}, \tag{3.20}$$

and the full, forced plus free, response of L can be expressed as

$$y_{[1,L]} = \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right] \begin{bmatrix} x_1 \\ u_{[1,L]} \end{bmatrix}. \tag{3.21}$$

Notice that

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]_{\text{z-s}} = \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right] \begin{bmatrix} 0_{n_x} \\ I_{Ln_u} \end{bmatrix} \tag{3.22}$$

and

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]_{\text{z-i}} = \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right] \begin{bmatrix} I_{n_x} \\ 0_{Ln_u} \end{bmatrix} \tag{3.23}$$

The same notation can be used to express various compositions of these operators. In each case the easiest way to verify these is to write out the state equations for one time step.

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]_{\text{z-s}} \left[\begin{array}{c|c} E_k & F_k \\ \hline G_k & H_k \end{array}\right]_{\text{z-s}} u = \left[\begin{array}{cc|c} A_k & B_kG_k & B_kH_k \\ 0 & E_k & F_k \\ \hline C_k & D_kG_k & D_kH_k \end{array}\right]_{\text{z-s}} u \tag{3.24}$$

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]_{\text{z-s}} \left[\begin{array}{c} E_k \\ \hline G_k \end{array}\right] x_1 = \left[\begin{array}{cc} A_k & B_kG_k \\ 0 & E_k \\ \hline C_k & D_kG_k \end{array}\right] \begin{bmatrix} 0 \\ I \end{bmatrix} x_1 \tag{3.25}$$

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]_{\text{z-s}} \left[\begin{array}{c|c} E_k & F_k \\ \hline G_k & H_k \end{array}\right] \begin{bmatrix} x_1 \\ u \end{bmatrix} = \left[\begin{array}{cc|c} A_k & B_kG_k & B_kH_k \\ 0 & E_k & F_k \\ \hline C_k & D_kG_k & D_kH_k \end{array}\right] \begin{bmatrix} 0 & 0 \\ I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x_1 \\ u \end{bmatrix} \tag{3.26}$$

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right] \left[\begin{array}{c} x_{11} \\ \left[\begin{array}{c|c} E_k & F_k \\ \hline G_k & H_k \end{array}\right] \begin{bmatrix} x_{21} \\ u \end{bmatrix} \end{array}\right] = \left[\begin{array}{cc|c} A_k & B_k G_k & B_k H_k \\ 0 & E_k & F_k \\ \hline C_k & D_k G_k & D_k H_k \end{array}\right] \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{21} \\ u \end{bmatrix} \tag{3.27}$$

Finally we consider state transformations. The following are identities.

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]_{\text{z-s}} = \left[\begin{array}{c|c} T_k A_k T_k^{-1} & T_k B_k \\ \hline C_k T_k^{-1} & D_k \end{array}\right]_{\text{z-s}} \tag{3.28}$$

$$\left[\begin{array}{c} A_k \\ \hline C_k \end{array}\right] = \left[\begin{array}{c} T_k A_k T_k^{-1} \\ \hline C_k T_k^{-1} \end{array}\right] T_1 \tag{3.29}$$

$$\left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right] = \left[\begin{array}{c|c} T_k A_k T_k^{-1} & T_k B_k \\ \hline C_k T_k^{-1} & D_k \end{array}\right] \begin{bmatrix} T_1 & 0 \\ 0 & I \end{bmatrix} \tag{3.30}$$

The representations in terms of the state-space $(T_k A_k T_k^{-1}, T_k B_k, C_k T_k^{-1}, D_k)$ are related to (A_k, B_k, C_k, D_k) by the state transformation $\tilde{x}_k = T_k x_k$, with T_k invertible for all k. These identities are useful because often, unobservable or uncontrollable dynamics are exposed with a suitable state transformation, due to the forms of $\tilde{A}_k = T_k A_k T_k^{-1}$, $\tilde{B}_k = T_k B_k$, and $\tilde{C}_k = C_k T_k^{-1}$. This can help simplify expressions. It's necessary to remember that the initial condition is affected by state transformation.

3.2.3 Computing a Particular Solution

In this section we discuss recursive, stable, efficient computation of the minimum-norm solution of problems like (3.13), when the matrices represent LTV systems. This enables us to find a particular solution of the linear dynamic system constraint for the unknown parts of x_1, e, and w.

To set the problem up, consider a system L with input u, output y, initial condition x_1, and state-space (A_k, B_k, C_k, D_k). Partition the input and initial condition into fixed parts $\bar{u}$ and $\bar{x}_1$, and free parts $\underline{u}$ and $\underline{x}_1$, respectively. Partition the state-space matrices accordingly, defining $\bar{B}_k$ and $\bar{D}_k$ as containing the columns of B_k and D_k that correspond to the fixed part of u, and $\underline{B}_k$ and $\underline{D}_k$ as containing the columns corresponding the the free parts. For instance this means

$$B_k u_k = \bar{B}_k \bar{u}_k + \underline{B}_k \underline{u}_k \quad \text{and} \quad D_k u_k = \bar{D}_k \bar{u}_k + \underline{D}_k \underline{u}_k.$$

Let $n_{\bar{x}_1}$ and $n_{\underline{x}_1}$ be the number of fixed and free components of the initial condition vector ($n_{\bar{x}_1} + n_{\underline{x}_1} = n_x$). Let $\bar{M}$ the $n_x \times n_{\bar{x}_1}$ matrix which "expands" the elements of the fixed initial conditions $\bar{x}_1$ into their places in x_1, and $\underline{M}$ the $n_x \times n_{\underline{x}_1}$ matrix that does the same for $\underline{x}_1$. For instance if $n_x = 3$, the first component of the initial condition is fixed and the last two are free, then

$$\bar{M} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \underline{M} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \text{and} \quad x_1 = \bar{M}\bar{x}_1 + \underline{M}\underline{x}_1. \tag{3.31}$$

Define

$$\bar{A}_k := \begin{cases} A_1\bar{M} & k = 1 \\ A_k & k > 1 \end{cases} \quad \text{and} \quad \bar{C}_k := \begin{cases} C_1\bar{M} & k = 1 \\ C_k & k > 1 \end{cases} \tag{3.32}$$

$$\underline{A}_k := \begin{cases} A_1\underline{M} & k = 1 \\ A_k & k > 1 \end{cases} \quad \text{and} \quad \underline{C}_k := \begin{cases} C_1\underline{M} & k = 1 \\ C_k & k > 1 \end{cases} \tag{3.33}$$

and similarly for $\bar{C}_k$ and $\underline{C}_k$. These definitions are for notational convenience in dealing with fixed vs free initial conditions. Note that for $k > 1$, $\bar{A}_k = \underline{A}_k = A_k$ and $\bar{C}_k = \underline{C}_k = C_k$. Only the $k = 1$ definitions are different; $\bar{A}_1$ and $\bar{C}_1$ contain the columns of A_1 and C_1 that correspond to the fixed part of the initial condition, and $\underline{A}_1$ and $\underline{C}_1$ contain the columns of A_1 and C_1 that correspond to the free part of the initial condition.

With these definitions, L can be partitioned into a part $\bar{L}$ that describes how the output depends on the fixed inputs and initial conditions, and another part $\underline{L}$ for the dependence on the free elements:

$$y = \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_c \end{array}\right] \begin{bmatrix} x_1 \\ u \end{bmatrix} \tag{3.34}$$

$$= \left[\begin{array}{c|c} A_k & \bar{B}_k \\ \hline C_k & \bar{D}_k \end{array}\right] \begin{bmatrix} \bar{M}\bar{x}_1 \\ \bar{u} \end{bmatrix} + \left[\begin{array}{c|c} A_k & \underline{B}_k \\ \hline C_k & \underline{D}_k \end{array}\right] \begin{bmatrix} \underline{M}\underline{x}_1 \\ \underline{u} \end{bmatrix} \tag{3.35}$$

$$= \underbrace{\left[\begin{array}{c|c} \bar{A}_k & \bar{B}_k \\ \hline \bar{C}_k & \bar{D}_k \end{array}\right]}_{=:\bar{L}} \begin{bmatrix} \bar{x}_1 \\ \bar{u} \end{bmatrix} + \underbrace{\left[\begin{array}{c|c} \underline{A}_k & \underline{B}_k \\ \hline \underline{C}_k & \underline{D}_k \end{array}\right]}_{=:\underline{L}} \begin{bmatrix} \underline{x}_1 \\ \underline{u} \end{bmatrix} \tag{3.36}$$

In this section the following minimum-norm problem is considered.

$$\begin{aligned} \underset{\underline{x}_1,\underline{u}}{\text{minimize}} \quad & \left\| \begin{pmatrix} \underline{x}_1 \\ \underline{u} \end{pmatrix} \right\|_2 \\ \text{subject to} \quad & y = \bar{L} \begin{bmatrix} \bar{x}_1 \\ \bar{u} \end{bmatrix} + \underline{L} \begin{bmatrix} \underline{x}_1 \\ \underline{u} \end{bmatrix}. \end{aligned} \tag{3.37}$$

Note that (3.13) has this form, and so in presenting a solution method for this problem, it can be readily applied to compute a feasible solution of the linear system equality constraint. The next section shows how to find a basis for the nullspace of $\underline{L}$.

The next four results are useful.

Theorem 3.1 (solution of minimum-norm problem) *Let L be an $n \times m$ matrix such that LL^* is invertible. Then for any y the system of equations $Lx = y$ has a feasible solution x, and*

$$\min_x \|x\|_2 \text{ subject to } Lx = y \tag{3.38}$$

has a unique optimal point

$$x^{mn} = L^*(LL^*)^{-1}y.$$

These are the so-called normal equations for this problem. For a proof and more discussion about this and other forms of the projection problem, see Luenberger [28].

Theorem 3.2 (minimum-norm solution) *Given L as in Theorem 3.1. If there exists an invertible K such that $(KL)(KL)^* = I$, then the solution of the minimum-norm problem (3.38) is given by*

$$x^{mn} = (KL)^*Ky.$$

proof Given such a K, define $Q := KL$. Then starting from Theorem 3.1 we have

$$\begin{aligned} x^{\text{mn}} &= L^*(LL^*)^{-1}y \\ &= Q^*K^{-*}(K^{-1}QQ^*K^{-*})^{-1}y \\ &= Q^*K^{-*}K^*Ky \\ &= (KL)^*Ky \end{aligned}$$

■

The following is useful when working with adjoints of linear operators derived from discrete-time dynamic systems.

Theorem 3.3 (adjoint of a certain linear operator) *Consider a linear operator L whose action $(x, u) \mapsto y$ is defined by*

$$y_k = C_k \tilde{x}_k + D_k u_k, \quad k = 1..L \tag{3.39}$$

with $\tilde{x}_k$ defined by

$$\tilde{x}_1 = x, \tag{3.40}$$

$$\tilde{x}_{k+1} = A_k \tilde{x}_k + B_k u_k, \quad k = 1..L-1 \tag{3.41}$$

for some compatibly-sized matrices $A_k, B_k, C_k, D_k, \quad k = 1..L$. (i.e. L is the full response of a discrete-time LTV system with state space (A_k, B_k, C_k, D_k).)

Then the adjoint operator $L^* : \psi \mapsto (\chi, \mu)$ is given by

$$\mu_k = B_k^* \tilde{\chi}_k + D_k^* \psi_k, \quad k = 1..L \tag{3.42}$$

$$\chi = \tilde{\chi}_0, \tag{3.43}$$

where $\tilde{\chi}_k$ is defined by

$$\tilde{\chi}_L = 0 \tag{3.44}$$

$$\tilde{\chi}_{k-1} = A_k^* \tilde{\chi}_k + C_k^* \psi_k, \quad k = 1..L. \tag{3.45}$$

Note that the definition of L^* evolves *backward* in time. One way to see the result is to write out the matrix associated with L as in (3.20), take its adjoint, and verify this agrees with the matrix representation of the operator L^* as defined. Or find another way to verify that

$$\langle \psi \, , \, \mathcal{L}(x, u) \rangle = \langle \mathcal{L}^* \psi \, , \, (x, u) \rangle \quad \forall \psi, x, u.$$

Also note that for LTI systems, if the forward recursion (3.41) is stable, i.e. A is Hurwitz, then the backward recursion (3.45) is also stable. ■

The last preliminary result is concerned with the inverse of a class of linear systems.

Theorem 3.4 (inverse of a class of linear dynamic system)

Let G be the zero-state response from time $t = 1$ to L of a discrete-time LTV system with state-space (A_k, B_k, C_k, D_k), with D_k square and invertible. Then G is invertible, and $G^{-1} = H$, where H is the zero-state response from time $t = 1$ to L of the discrete-time LTV system with state-space $(A_k - B_k D_k^{-1} C_k, B_k D_k^{-1}, -D_k^{-1} C_k, D_k^{-1})$.

proof: G has the matrix representation

$$G: \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right]_{\text{z-s}},$$

and H has the matrix representation

$$H: \left[\begin{array}{c|c} A_k - B_k D_k^{-1} C_k & B_k D_k^{-1} \\ \hline -D_k^{-1} C_k & D_k^{-1} \end{array}\right]_{\text{z-s}}.$$

The composition of H with G therefore has matrix representation

$$HG: \left[\begin{array}{cc|c} A_k - B_k D_k^{-1} C_k & B_k D_k^{-1} C_k & B_k D_k^{-1} D_k \\ 0 & A_k & B_k \\ \hline -D_k^{-1} C_k & D_k^{-1} C_k & D_k^{-1} D_k \end{array}\right]_{\text{z-s}}$$

(see (3.24)). Partition the state of HG as (x_G, x_H) and considering a state transformation

$$\begin{bmatrix} \tilde{x}_G \\ \tilde{x}_H \end{bmatrix} = \begin{bmatrix} I & -I \\ 0 & I \end{bmatrix} \begin{bmatrix} x_G \\ x_H \end{bmatrix},$$

we see

$$HG: \left[\begin{array}{cc|c} A_k - B_k D_k^{-1} C_k & 0 & 0 \\ 0 & A_k & B_k \\ \hline -D_k^{-1} C_k & 0 & I \end{array}\right]_{\text{z-s}} = \left[\begin{array}{c|c} A_k - B_k D_k^{-1} C_k & 0 \\ \hline -D_k^{-1} C_k & I \end{array}\right]_{\text{z-s}}.$$

The first equality is like (3.28), and the second equality comes from eliminating unobservable states. It is clear that the expression on the right hand side is the identity matrix; the input has no effect on the output and the feedthrough term is identity. Or look at (3.15). This shows that H is a left inverse of G. It can be shown to be a right inverse in a similar way.

remark 1: A quick way to see where the state space of H comes from. The dynamic system of G evolves according to:

$$\begin{bmatrix} x_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} A_k & B_k \\ C_k & D_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix}. \tag{3.46}$$

Since D is invertible we can algebraically manipulate this into

$$\begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} = \begin{bmatrix} A_k - B_k D_k^{-1} C_k & B_k D_k^{-1} \\ -D_k^{-1} C_k & D_k^{-1} \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix}, \tag{3.47}$$

and given the output $y_k, k = 1..l$ of G and using $x_1 = 0$ since y is the zero-state response of G, we can run the latter system to compute the input $u_k, k = 1..L$ of G.

■

A main assumption we make in this and the following section is that the "D" matrix of $\underline{L}$, $\underline{D}_k$, is such that $\underline{D}_k\underline{D}_k^*$ is invertible for all k. As suggested in Section 3.2.1, this is always the case when the model includes noise on all measurements. It will be shown that this implies $\underline{LL}^*$ is invertible as well. In this case according to Theorem 3.1 the min-norm solution of (3.37) is given by

$$\begin{bmatrix} \underline{x}_1{}^{\mathrm{mn}} \\ \underline{u}^{\mathrm{mn}} \end{bmatrix} = \underline{L}^*(\underline{LL}^*)^{-1}\bar{y} \tag{3.48}$$

where

$$\bar{y} := y - \bar{L}\begin{bmatrix} \bar{x}_1 \\ \bar{u} \end{bmatrix}. \tag{3.49}$$

Now, given $\underline{L}$ we know how to compute $\underline{L}^*$ and apply its operation to an input vector in a recursive manner using Theorem 3.3. And according to Theorem 3.4 we can represent the inverse of the zero-state response operator for a dynamic system by a recursive operator. However we cannot actually compute $(\underline{LL}^*)^{-1}$ recursively since we don't have a convenient representation of $\underline{LL}^*$ for doing so. We have state space representations of $\underline{L}$ and $\underline{L}^*$, but we don't have a state space representation for the composition; it's not straightforward since $\underline{L}$ evolves forward in time while $\underline{L}^*$ goes backward.

Also keep in mind $\bar{y}$ isn't directly known, rather it has to be computed from knowledge of y, $\bar{x}_1$, $\bar{u}$, and L as in (3.49). In particular the operation

$$\bar{L}\begin{bmatrix} \bar{x}_1 \\ \bar{u} \end{bmatrix},$$

might be problematic as we haven't made the assumption that L is a stable system.

But we can set things up so that we can use a solution method motivated by Theorem 3.2, one which is possible to compute recursively, and does not require simulations

that are potentially unstable. To this end, define

$$K_1 := \left[\begin{array}{c} A_k + K_k C_k \\ \hline T_k^{-1/2} C_k \end{array}\right] \tag{3.50}$$

$$K_u := \left[\begin{array}{c|c} A_k + K_k C_k & \bar{B}_k + K_k \bar{D}_k \\ \hline T_k^{-1/2} C_k & T_k^{-1/2} \bar{D}_k \end{array}\right]_{\text{z-s}} \tag{3.51}$$

$$K_y := \left[\begin{array}{c|c} A_k + K_k C_k & -K_k \\ \hline T_k^{-1/2} C_k & -T_k^{-1/2} \end{array}\right]_{\text{z-s}} \tag{3.52}$$

where K_k and T_k are defined by the Kalman filtering equations

$$P_1 := I \tag{3.53}$$

$$\begin{bmatrix} R_k & S_k \\ {S_k}^* & T_k \end{bmatrix} := \begin{bmatrix} \underline{A}_k & \underline{B}_k \\ \underline{C}_k & \underline{D}_k \end{bmatrix} \begin{bmatrix} P_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \underline{A}_k & \underline{B}_k \\ \underline{C}_k & \underline{D}_k \end{bmatrix}^* \tag{3.54}$$

$$= \begin{bmatrix} \underline{A}_k P_k {\underline{A}_k}^* + \underline{B}_k \underline{B}_k^* & \underline{A}_k P_k {\underline{C}_k}^* + \underline{B}_k \underline{D}_k^* \\ (\underline{A}_k P_k {\underline{C}_k}^* + \underline{B}_k \underline{D}_k^*)^* & \underline{C}_k P_k {\underline{C}_k}^* + \underline{D}_k \underline{D}_k^* \end{bmatrix} \quad k = 1..L \tag{3.55}$$

$$P_{k+1} := R_k - S_k {T_k}^{-1} {S_k}^* \quad k = 1..L. \tag{3.56}$$

$$K_k := -S_k {T_k}^{-1} \quad k = 1..L \tag{3.57}$$

First note that $\underline{D}_k \underline{D}_k^*$ is invertible by assumption i.e. $\underline{D}_k \underline{D}_k^* \succ 0$, which implies that $T_k = \underline{C}_k P_k {\underline{C}_k}^* + \underline{D}_k \underline{D}_k^* \succ 0$ and is invertible as well. Therefore definitions (3.50)–(3.52) are valid.

Toward applying Theorem 3.2, notice that K_y is invertible by Theorem 3.4. Next it can be shown that $(K_y \underline{L})(K_y \underline{L})^* = I$ by demonstrating $(K_y \underline{L})(K_y \underline{L})^* \psi = \psi$ for all ψ.

First compute a state-space representation of the composition

$$
\begin{aligned}
Q &:= K_y \underline{L} && (3.58)\\
&= \left[\begin{array}{c|c} \underline{A}_k + K_k\underline{C}_k & -K_k \\ \hline T_k^{-1/2}\underline{C}_k & -T_k^{-1/2} \end{array}\right]_{\text{z-s}} \left[\begin{array}{c|c} \underline{A}_k & \underline{B}_k \\ \hline \underline{C}_k & \underline{D}_k \end{array}\right] && (3.59)\\
&= \left[\begin{array}{cc|c} \underline{A}_k + K_k\underline{C}_k & -K_k\underline{C}_k & -K_k\underline{D}_k \\ 0 & \underline{A}_k & \underline{B}_k \\ \hline T_k^{-1/2}\underline{C}_k & -T_k^{-1/2}\underline{C}_k & -T_k^{-1/2}\underline{D}_k \end{array}\right] \begin{bmatrix} 0 & 0 \\ I & 0 \\ 0 & I \end{bmatrix} && (3.60)\\
&= \left[\begin{array}{cc|c} \underline{A}_k + K_k\underline{C}_k & 0 & \underline{B}_k + K_k\underline{D}_k \\ 0 & \underline{A}_k & \underline{B}_k \\ \hline -T_k^{-1/2}\underline{C}_k & 0 & -T_k^{-1/2}\underline{D}_k \end{array}\right] \begin{bmatrix} -I & I & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} 0 & 0 \\ I & 0 \\ 0 & I \end{bmatrix} && (3.61)\\
&= \left[\begin{array}{c|c} \underline{A}_k + K_k\underline{C}_k & \underline{B}_k + K_k\underline{D}_k \\ \hline -T_k^{-1/2}\underline{C}_k & -T_k^{-1/2}\underline{D}_k \end{array}\right] && (3.62)
\end{aligned}
$$

where in going from the first to the second line we take advantage of the fact that the zero-state response doesn't depend on the A or C matrix at time $t = 1$ and we can substitute anything, in going from the third to the fourth line we applied a state transformation $\tilde{x} = \begin{bmatrix} -I & I \\ 0 & I \end{bmatrix} x$, and in going from the fourth to the fifth we eliminated the second set of states as they are unobservable and don't affect the output.

Now suppose we drive the system Q by the output of $Q^*\psi$. Refer to Theorem 3.3 to see how Q^* is defined. Letting u, y, x be the input, output, and state of Q, then Q evolves according to

$$\begin{bmatrix} x_{k+1} \\ y_k \end{bmatrix} = Q_k \begin{bmatrix} x_k \\ u_k \end{bmatrix} \tag{3.63}$$

and letting ψ, μ, χ be the input, output, and state of Q^*, then Q^* evolves according to

$$\begin{bmatrix} \chi_{k-1} \\ \mu_k \end{bmatrix} = Q_k^* \begin{bmatrix} \chi_k \\ \psi_k \end{bmatrix}, \tag{3.64}$$

where

$$Q_k := \begin{bmatrix} \underline{A}_k + K_k\underline{C}_k & \underline{B}_k + K_k\underline{D}_k \\ -T_k^{-1/2}\underline{C}_k & -T_k^{-1/2}\underline{D}_k \end{bmatrix} \tag{3.65}$$

as suggested by (3.62).

With this setup,

Theorem 3.5 *Given an input vector ψ, when Q is driven by the output of $Q^*\psi$ to produce y, then*

$$x_k = P_k \chi_{k-1} \quad \text{and} \quad y_k = \psi_k, \quad k = 1..L. \tag{3.66}$$

proof First note that part of the output vector of Q^* is χ_0, which becomes the initial condition x_1 of Q. Thus $x_1 = P_1\chi_0$, since $P_1 = I$.

Next, notice that

$$Q_k = \begin{bmatrix} I & 0 \\ 0 & T_k^{-1/2} \end{bmatrix} \begin{bmatrix} I & K_k \\ 0 & -I \end{bmatrix} \begin{bmatrix} \underline{A}_k & \underline{B}_k \\ \underline{C}_k & \underline{D}_k \end{bmatrix}$$

and then that

$$\begin{aligned} Q_k \begin{bmatrix} P_k & 0 \\ 0 & I \end{bmatrix} Q_k^* &= \begin{bmatrix} I & 0 \\ 0 & T_k^{-1/2} \end{bmatrix} \begin{bmatrix} I & K_k \\ 0 & -I \end{bmatrix} \begin{bmatrix} \underline{A}_k & \underline{B}_k \\ \underline{C}_k & \underline{D}_k \end{bmatrix} \begin{bmatrix} P_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \underline{A}_k & \underline{B}_k \\ \underline{C}_k & \underline{D}_k \end{bmatrix}^* \begin{bmatrix} I & K_k \\ 0 & -I \end{bmatrix}^* \begin{bmatrix} I & 0 \\ 0 & T_k^{-1/2} \end{bmatrix}^* \\ &= \begin{bmatrix} I & 0 \\ 0 & T_k^{-1/2} \end{bmatrix} \begin{bmatrix} I & -S_k T_k^{-1} \\ 0 & -I \end{bmatrix} \begin{bmatrix} R_k & S_k \\ S_k{}^* & T_k \end{bmatrix} \begin{bmatrix} I & -S_k T_k^{-1} \\ 0 & -I \end{bmatrix}^* \begin{bmatrix} I & 0 \\ 0 & T_k^{-1/2} \end{bmatrix}^* \\ &= \begin{bmatrix} I & 0 \\ 0 & T_k^{-1/2} \end{bmatrix} \begin{bmatrix} P_{k+1} & 0 \\ 0 & T_k \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & T_k^{-1/2} \end{bmatrix}^* \\ &= \begin{bmatrix} P_{k+1} & 0 \\ 0 & I \end{bmatrix} \quad k = 1..L \end{aligned}$$

where we used the definitions of R_k, S_k, T_k, K_k, and P_k in (3.54)–(3.53). This makes the following easy to see. Suppose, for $1 \le k \le L$, that indeed $x_k = P_k\mu_{k-1}$. Then

$$\begin{bmatrix} x_{k+1} \\ y_k \end{bmatrix} = Q_k \begin{bmatrix} x_k = P_k\chi_{k-1} \\ \mu_k \end{bmatrix} = Q_k \begin{bmatrix} P_k & 0 \\ 0 & I \end{bmatrix} Q_k^* \begin{bmatrix} \chi_k \\ \psi_k \end{bmatrix} = \begin{bmatrix} P_{k+1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \chi_k \\ \psi_k \end{bmatrix}. \tag{3.67}$$

This completes an inductive argument which shows $x_k = P_k\chi_{k-1},\ k = 1..L$, and in the process we have shown that this implies $y_k = \psi_k,\ k = 1..L$, also. ■

This shows that $(K_y\underline{L})(K_y\underline{L})^* = I$, and since K_y is invertible, it follows from Theorem 3.2 that the minimum-norm solution of (3.37) is given by

$$\begin{bmatrix} \underline{x}_1{}^{\text{mn}} \\ \underline{u}^{\text{mn}} \end{bmatrix} = (K_y\underline{L})^* K_y \bar{y}. \tag{3.68}$$

When working with LTI systems and assuming $(\underline{A}, \underline{C})$ is observable, the operator $(K_y\underline{L})^*$ in (3.68) corresponds to a stable dynamic system. First, Q as defined in in (3.62) is

stable. This is a property of the Kalman filtering equations (3.50)–(3.52). Then, Q stable implies Q^* is also stable (see Theorem 3.3), and so its action can be computed in a stable recursive way. Now it remains to compute $K_y\bar{y}$.

Recall that in addition to K_y we defined a K_1 and K_u in (3.50) and (3.51). By assumption the signals y and $\bar{u}$ are known. Observe:

$$K_1\bar{M}\bar{x}_1 + K_u\bar{u} + K_y y = \begin{bmatrix} K_1 & K_u \end{bmatrix} \begin{bmatrix} \bar{M} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{u} \end{bmatrix} + K_y\bar{L} \begin{bmatrix} \bar{x}_1 \\ \bar{u} \end{bmatrix} + K_y\underline{L} \begin{bmatrix} \underline{x}_1 \\ \underline{u} \end{bmatrix} \tag{3.69}$$

$$= K_y\underline{L} \begin{bmatrix} \underline{x}_1 \\ \underline{u} \end{bmatrix} \tag{3.70}$$

$$= K_y\bar{y}, \tag{3.71}$$

where in going from (3.69) to (3.70) we used that the first two terms cancel, due to:

$$\begin{bmatrix} K_1 & K_u \end{bmatrix} \begin{bmatrix} \bar{M} & 0 \\ 0 & I \end{bmatrix} = \left[\begin{array}{c|c} A_k + K_kC_k & \bar{B}_k + K_k\bar{D}_k \\ \hline T_k^{-1/2}C_k & T_k^{-1/2}\bar{D}_k \end{array}\right] \begin{bmatrix} \bar{M} & 0 \\ 0 & I \end{bmatrix} \tag{3.72}$$

$$= \left[\begin{array}{c|c} \bar{A}_k + K_k\bar{C}_k & -\bar{B}_k - K_k\bar{D}_k \\ \hline T_k^{-1/2}\bar{C}_k & -T_k^{-1/2}\bar{D}_k \end{array}\right] \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}, \tag{3.73}$$

and

$$K_y\bar{L} = \left[\begin{array}{c|c} A_k + K_kC_k & -K_k \\ \hline T_k^{-1/2}C_k & -T_k^{-1/2} \end{array}\right]_{\text{z-s}} \left[\begin{array}{c|c} \bar{A}_k & \bar{B}_k \\ \hline \bar{C}_k & \bar{D}_k \end{array}\right] \tag{3.74}$$

$$= \left[\begin{array}{c|c} \bar{A}_k + K_k\bar{C}_k & -\bar{B}_k - K_k\bar{D}_k \\ \hline T_k^{-1/2}\bar{C}_k & -T_k^{-1/2}\bar{D}_k \end{array}\right] \begin{bmatrix} -I & 0 \\ 0 & I \end{bmatrix}, \tag{3.75}$$

by a development similar to that used in (3.58)–(3.62). Since K_1, K_u, and K_y are stable systems and $\bar{x}_1$, $\bar{u}$, and y are known, we are able to compute $K_y\bar{y}$ stably as in (3.69).

End-to-end we have:

$$\begin{bmatrix} \underline{x}_1{}^{\mathrm{mn}} \\ \underline{u}^{\mathrm{mn}} \end{bmatrix} = (K_y \underline{L})^* K_y \bar{y} \tag{3.76}$$

$$= (K_y \underline{L})^* \left(K_1 \bar{M} \bar{x}_1 + K_u \bar{u} + K_y y \right) \tag{3.77}$$

$$= \left[\begin{array}{c|c} \underline{A}_k + K_k \underline{C}_k & \underline{B}_k + K_k \underline{D}_k \\ \hline -T_k^{-1/2} \underline{C}_k & -T_k^{-1/2} \underline{D}_k \end{array} \right]^* \left[\begin{array}{c|cc} \bar{A}_k + K_k \bar{C}_k & \bar{B}_k + K_k \bar{D}_k & -K_k \\ \hline T_k^{-1/2} \bar{C}_k & T_k^{-1/2} \bar{D}_k & -T_k^{-1/2} \end{array} \right] \begin{pmatrix} \bar{x}_1 \\ \bar{u} \\ y \end{pmatrix} \tag{3.78}$$

3.2.4 Computing a Nullspace Basis

In this section the concern is to find a basis for the nullspace of

$$\underline{L} = \left[\begin{array}{c|c} \underline{A}_k & \underline{B}_k \\ \hline \underline{C}_k & \underline{D}_k \end{array} \right]. \tag{3.79}$$

When this system represents the dependence of the output of $\mathcal{L}$ on unknown inputs and initial conditions, this is the second step in finding a parametrization of the feasible set of the linear equality constraints as described in Section 3.2.

By a "basis" of the nullspace, we mean an operator whose range is equal to the nullspace of $\underline{L}$, and whose own nullspace is the trivial nullspace.

The following 2 results are useful here.

Theorem 3.6 (nullspace basis of a finite-rank surjective linear map) *Suppose L is a finite-rank linear operator of the form*

$$L((x, y)) := L_1 x + L_2 y,$$

with $L_1 : \mathbf{R}^n \to \mathbf{R}^n$ an invertible linear operator and $L_2 : \mathbf{R}^p \to \mathbf{R}^n$. Define $M : \mathbf{R}^p \to \mathbf{R}^{n+p}$ by

$$M(f) := (-L_1^{-1} L_2 f, f). \tag{3.80}$$

Then M is a basis for the nullspace of L, i.e.

$$\mathcal{R}(M) = \mathcal{N}(L)$$

and

$$\mathcal{N}(M) = \{0\},$$

i.e. M is a bijection between $\mathbf{R}^p$ and $\mathcal{N}(L)$.

proof Suppose $(x, y) \in \mathcal{R}(M)$. Then $\exists f$ with $y = f$ and $x = -L_1^{-1}L_2 f$, and so $L((x, y)) = L_1(-L_1^{-1}L_2 f) + L_2 f = -L_2 f + L_2 f = 0$. Conversely if $(x, y) \in \mathcal{N}(L)$ this means $L_1 x + L_2 y = 0$ which implies $L_1 x = -L_2 y$ which implies $x = -L_1^{-1}L_2 y$ which means $(x, y) = M(y)$, i.e. $(x, y) \in \mathcal{R}(M)$. We have shown $\mathcal{R}(M) = \mathcal{N}(L)$. That M has a trivial nullspace is evident from its definition. ■

Note that any finite-rank surjective linear map can be put into the form of the L in this theorem with a suitable invertible transformation of the input. Suppose $L : \mathbf{R}^m \to \mathbf{R}^n$ is a linear map of rank n. Then there exists an input transformation V so that with V partitioned into its first n and remaining $m - n$ inputs as $V = [V_1\ V_2]$,

$$LVx \quad = \quad (\tilde{L}_1 := LV_1)x_1 + (\tilde{L}_2 := LV_2)x_2 \quad = \quad \begin{bmatrix} \tilde{L}_1 & \tilde{L}_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

and $\tilde{L}_1$ is invertible. A suitable input transformation can be seen by computing the singular value decomposition of L or via a Graham-Schmidt orthogonalization procedure, to name two possibilities.

The next result shows how, given a basis of the nullspace of an operator like $U^{-1}AV$ where U and V are invertible transformations, how to adjust it to be the basis of the nullspace of A.

Theorem 3.7 (nullspace under invertible input and output transformations) *Given U, V invertible. Suppose*

$$\mathcal{R}(B) = \mathcal{N}(U^{-1}AV).$$

Then

$$\mathcal{R}(VB) = \mathcal{N}(A).$$

proof First, $\mathcal{N}(U^{-1}AV) = \mathcal{N}(AV)$ since $Ux \neq 0$ if and only if $x \neq 0$.

$\supseteq$ Suppose $x \in \mathcal{R}(VB)$. Then $\exists w$ such that $x = VBw$, which implies $Ax = AVBw = 0$ since $Bw \in \mathcal{R}(B)$.

$\subseteq$ Suppose $x \in \mathcal{N}(A)$. Then $AVV^{-1}x = 0$, which by the hypothesis means $V^{-1}x \in \mathcal{R}(B)$, or $V^{-1}x = Bw$ for some w, or $x = VBw$.

■

Now apply these results to the problem at hand. In this case the linear operator whose nullspace we seek is given in (3.79). To get it into a form to apply Theorem 3.6, first compute the SVD of the $\underline{D}_k$:

$$\underline{D}_k = U_k \begin{bmatrix} \Sigma_k & 0 \end{bmatrix} V_k^*. \tag{3.81}$$

In general $\underline{D}_k$ is $m \times r$. Our main assumption in this section, same as was assumed when computing the particular solution in the previous section, is that $\underline{D}_k$ has rank m, i.e. it is full row rank, and therefore Σ_k, which is square $m \times m$, is invertible. Partition V into the first m and last $r - m$ columns:

$$V_k = \begin{bmatrix} V_{1,k} & V_{2,k} \end{bmatrix}. \tag{3.82}$$

Define a transformed version $\tilde{\underline{L}}$ of $\underline{L}$ by applying V_k to the input at time k, running $\underline{L}$ with this transformed input and the initial condition, and then multiplying the output at time k by U_k^* to produce $\tilde{\underline{L}}$'s output. These are invertible input and output transformations. Partitioning the input to $\tilde{\underline{L}}$ into the first m and remaining $r - m$ inputs,

$$\tilde{\underline{L}} \;:=\; \operatorname{diag}\left(\{U_k^*\}\right) \; \underline{L} \begin{bmatrix} I & 0 & 0 \\ 0 & \operatorname{diag}(\{V_{1,k}\}) & \operatorname{diag}(\{V_{2,k}\}) \end{bmatrix} \;=\; \begin{bmatrix} L_0 & L_1 & L_2 \end{bmatrix} \tag{3.83}$$

where we have defined

$$\operatorname{diag}(\{V_{i,k}\}) := \begin{bmatrix} V_{i,1} & & \\ & \ddots & \\ & & V_{i,L} \end{bmatrix} \tag{3.84}$$

$$L_0 := \left[\begin{array}{c} \underline{A}_k \\ \hline U_k^* \underline{C}_k \end{array}\right] \tag{3.85}$$

$$L_1 := \left[\begin{array}{c|c} \underline{A}_k & \underline{B}_k V_{1,k} \\ \hline U_k^* \underline{C}_k & \Sigma_k \end{array}\right]_{\text{z-s}} \tag{3.86}$$

$$L_2 := \left[\begin{array}{c|c} \underline{A}_k & \underline{B}_k V_{2,k} \\ \hline U_k^* \underline{C}_k & 0 \end{array}\right]_{\text{z-s}}. \tag{3.87}$$

Since we assumed that Σ_k is invertible for all k, then L_1 is an invertible operator,

according to Theorem 3.4. Applying Theorem 3.6,

$$\mathcal{N}(\tilde{\underline{L}}) = \mathcal{R}\left(\begin{bmatrix} I & 0 \\ -L_1^{-1}L_0 & -L_1^{-1}L_2 \\ 0 & I \end{bmatrix}\right) \tag{3.88}$$

$$= \left\{\left(x,\ -L_1^{-1}\left[L_0 x + L_2 f\right],\ f\right) : x \in \mathbf{R}^{n_{\underline{x}_1}},\ f \in \mathbf{R}^{L(n_{\underline{u}} - n_y)}\right\} \tag{3.89}$$

Composing the definitions (3.85)–(3.87), using Theorem 3.4, and after a state transformation and eliminating unobservable states, we have

$$L_1^{-1}\begin{bmatrix} L_0 & L_2 \end{bmatrix} = \left[\begin{array}{c|c} \underline{A}_k - \underline{B}_k V_{1,k}\Sigma_k^{-1}U_k^*\underline{C}_k & \underline{B}_k V_{2,k} \\ \hline \Sigma_k^{-1}U_k^*\underline{C}_k & 0 \end{array}\right] \tag{3.90}$$

We now have a representation for a basis of the nullspace of $\tilde{\underline{L}}$, (which can be related to the nullspace of $\underline{L}$ as in Theorem 3.7). This representation is related to a dynamic system and can be computed recursively; instead of doing a large matrix multiplication to generate an element of the nullspace as a linear combination of basis vectors, the element can be computed as the output of a dynamic system, whose input represents the coefficients of the linear combination.

In general there is no guarantee that the system (3.90) is well-behaved; in particular it may be unstable. However it's possible to use this as a starting point to find another basis which *does* involve stable systems.

Notice that what the parametrization (3.89) does is express the nullspace of $\tilde{\underline{L}}$ as the set of all input-output pairs of the system $-L_1^{-1}[L_0\ L_2]$. The following theorem shows how to find other ways to parametrize the input-output pairs of a given linear system.

Theorem 3.8 (input-output pairs) *Suppose we have a system with matrix representation*

$$G = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right].$$

Define

$$N := \left[\begin{array}{c|c} A + BF & B \\ \hline C + DF & D \end{array}\right] \tag{3.91}$$

$$D := \left[\begin{array}{c|c} A + BF & B \\ \hline F & I \end{array}\right], \tag{3.92}$$

where F is any compatibly-sized matrix. Consider the operators

$$B_1(x, f) := \big(x, f, G(x, f)\big) \tag{3.93}$$

$$B_2(\mu, w) := \big(\mu, D(\mu, w), N(\mu, w)\big). \tag{3.94}$$

Then

$$\mathcal{R}(B_1) = \mathcal{R}(B_2).$$

proof

$\subseteq$ First, motivated by Theorem 3.4 we define

$$H := \left[\begin{array}{c|c} A & B \\ \hline -F & I \end{array}\right]. \tag{3.95}$$

Now given some x and f, choose $\mu = x$, and define

$$w := H \begin{pmatrix} x \\ f \end{pmatrix}.$$

Then $N(\mu, w) = N(x, H(x, f)) = G(x, f)$ and $D(\mu, w) = D(x, H(x, f)) = f$. (For the latter, consider the initial condition and input responses of D and H separately, and use Theorem 3.4.) Thus the range of B_1 is contained in the range of B_2.

$\supseteq$ Given some μ and w. Define $x = \mu$ and $f = D(\mu, w)$, and observe

$$G\begin{pmatrix} x \\ f \end{pmatrix} = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \left[\begin{array}{c|c} A+BF & B \\ \hline F & I \end{array}\right] \begin{pmatrix} x \\ x \\ w \end{pmatrix} \tag{3.96}$$

$$= \left[\begin{array}{cc|c} A & BF & B \\ 0 & A+BF & B \\ \hline C & DF & D \end{array}\right] \begin{pmatrix} x \\ x \\ w \end{pmatrix} \tag{3.97}$$

$$= \left[\begin{array}{cc|c} A & 0 & 0 \\ 0 & A+BF & B \\ \hline C & C+DF & D \end{array}\right] \begin{bmatrix} I & -I \\ 0 & I \end{bmatrix} \begin{pmatrix} x \\ x \\ w \end{pmatrix} \tag{3.98}$$

$$= \left[\begin{array}{c|c} A+BF & B \\ \hline C+DF & D \end{array}\right] \begin{pmatrix} x \\ w \end{pmatrix} \tag{3.99}$$

$$= N(\mu, w), \tag{3.100}$$

where in going from (3.97) to (3.98) we used the state transformation $T = \begin{bmatrix} I & -I \\ 0 & I \end{bmatrix}$, and in going to (3.99) we eliminated some uncontrollable states which are identically zero. This shows that the range of B_2 is contained in the range of B_1.

■

So, given a system G, one way to generate its input-output pairs is to choose inputs and simulate G for those inputs to generate the outputs. In this theorem, for any F we choose, we get a new way to generate input-output pairs of G which involves simulating the systems N and D, both of which have dynamics governed by $(A+BF)$. When (A, B) is stabilizable we can choose F such that $A+BF$ has stable eigenvalues. With such a choice, (3.91) and (3.92) form a right *coprime factorization* of G. Note that

$$N\begin{pmatrix} x_N \\ D^{-1}(x_D, f) \end{pmatrix} = \left[\begin{array}{cc|c} A+BF & 0 & 0 \\ 0 & A & B \\ \hline C+DF & C & D \end{array}\right] \begin{pmatrix} x_N - x_D \\ x_D \\ f \end{pmatrix}, \tag{3.101}$$

which equals $G(x, f)$ if $x_N = x_D = x$, so in a sense $G = ND^{-1}$. See Zhou et al. [46] for more about discrete-time coprime factorizations. For time-varying systems, add a subscript to all the matrices to get the desired result.

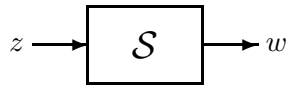


Figure 3.2: Static operator

Putting it together: we want a basis for the nullspace of $\underline{L}$. We can transform $\underline{L}$ into an operator $\underline{\tilde{L}}$ of the form for Theorem 3.6 using suitable unitary input and output transformations. Theorem 3.6 then provides a system representation of a basis for the nullspace of $\underline{\tilde{L}}$, which can easily be adjusted to get the nullspace basis of $\underline{L}$ using Theorem 3.7. The elements in this basis are pairs of the form of a free vector and the response of a certain system to this vector. Finally, we can use a coprime factorization of the system to produce these input-output pairs in a numerically well-behaved manner. This last step assumes that $L_1^{-1}[L_0 \; L_2]$ as defined in (3.85)–(3.87) is stabilizable.

3.3 Staticness Criteria, and Other Properties of $\mathcal{S}$

Next we concentrate on knowledge and assumptions associated with the static operator $\mathcal{S}$. This element with its inputs and outputs is isolated in Figure 3.2. In the framework we have laid out this means we would like to develop criteria for judging whether, for a particular pair of finite-length signals (z, w), an operator which fits the profile of $\mathcal{S}$ could map z to w. The less likely it seems that this could be true, the less willing we are to accept (z, w) as estimates of the input and output signals of $\mathcal{S}$ that were realized during the experiment.

Recall the definition of a static operator: in the context of dynamic systems, an operator is static if the output at time t depends only on the value of the input at time t. It does not depend on previous or future inputs, or time. In this case there should appear to be a single-valued function which relates inputs to outputs.

This leads to the following preliminary idea. We do not allow input/output estimates for which a particular value of the input is repeated at two distinct times t_1 and t_2, that is $z(t_1) = z(t_2)$, but for which $w(t_1) \neq w(t_2)$. A static operator could not map such a z to such a w because in this case there is no single-valued function S such that $w(t) = S(z(t)) \;\; \forall t$.

However this criterion turns out to not be especially useful in practice. The prob-

lem is that the set of finite data records with repeated inputs like this is very small; indeed when considered as a subset of all possible records of length L, this subset has zero measure. (For instance consider SISO data records of length 2. Then the input record belongs to $\mathbf{R}^2$. The subset of $\mathbf{R}^2$ with the two inputs being equal is the one-dimensional subspace $z_1 = z_2$ which as a subset of $\mathbf{R}^2$ has zero area.) Even for an estimate which *is* ruled out by this criterion, there is another lying arbitrarily close (in any norm) that isn't ruled out, by slightly perturbing the repeated input points so that they are unique and there are no longer repeated inputs. The modified input-output pair passes the test and is not at odds with a static relationship, although for all practical purposes it may be indistinguishable from one that does not pass the test.

There might be some hope to arrange so that the particular input applied to $\mathcal{S}$ during the experiment is known and *does* have repeated inputs. This of course is not possible in every situation, as sometimes we don't have direct control over $\mathcal{S}$'s input. Further, even if we do have control authority over the input there is the problem of dealing with unknown disturbances which disrupt our perfectly planned experiment containing repeated inputs. Alternatively if we can directly sense the input there is the problem of measurement noise, so that even if our sensors tell us an input value was repeated, we can't be sure.

Conversely if a data record does *not* have repeated inputs then it is always possible to find a static operator relating the two. Simply fill in the missing points of the function's domain with any value and it's a static map. For example it's possible to find an infinitely differentiable function to fit the data by interpolating the points with a polynomial. When the input is scalar perhaps the simplest choice is to use linear interpolation between the specified points, and in higher dimensions one could extend this idea. In any case we are able to rule out very few estimates this way. As for the rest one is just as good as another; the repeated-input criterion does not differentiate between them at all.

Even though a particular finite-length input-output data record might not be technically inconsistent with a static input-output relationship, it is often the case that upon viewing the data one is able to intuitively make the assessment that a static relationship is unlikely. Consider the example shown in Figure 3.3 of two SISO systems driven by the same input z to produce an output w. The first system is dynamic (discrete-time, first order, stable), and the second is a static function. Looking at the scatter plot of w versus z, and the corresponding function estimate formed by linear interpolation of those points, it seems clear that the relationship between z and w in the first case is not static, while in

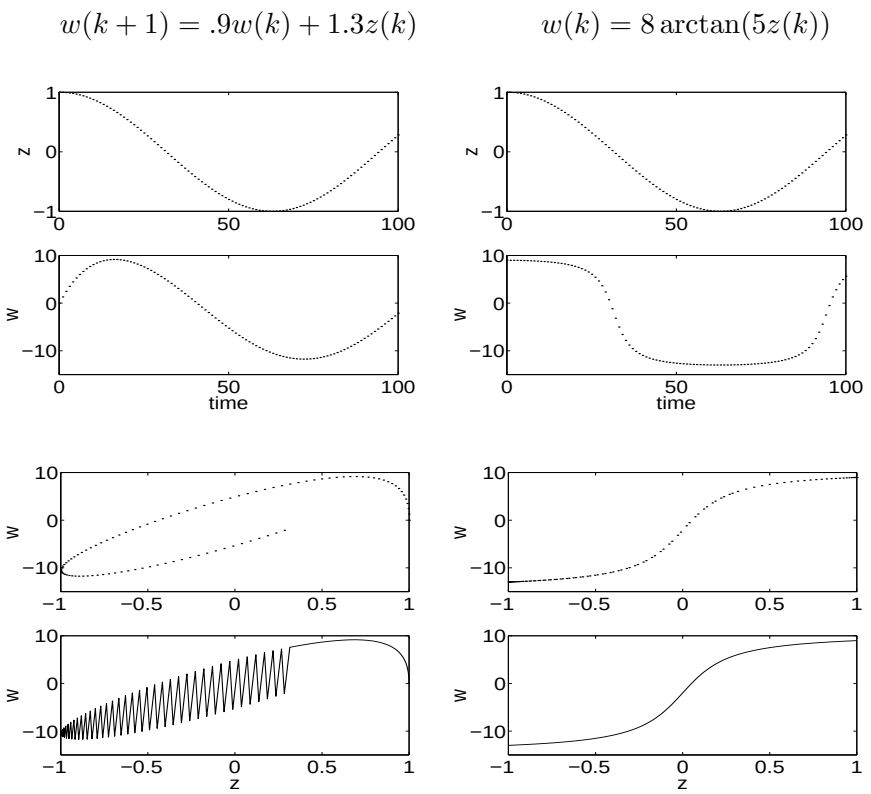


Figure 3.3: Two systems driven by the same input: looking at the scatter plot from a dynamic vs. a static system

the second case it appears to indeed be static, or at the least we would not rule it out. But the simple repeated-input criteria does not allow us to rule out the data record from the non-static system. The question is, how can we measure or enforce what we intuitively see?

Generally speaking our intuition is that the scatter plot of input-output pairs coming from a static operator should be smooth, and not jumpy. Here "smooth" does not refer to a technical definition in terms of differentiability, but a more intuitive notion of not oscillating wildly. Although one should keep in mind that this is certainly not the case in general, for many static operators of practical interest this intuition is reasonable. Ultimately we need concrete quantitative cost functions that both capture our intuitive notions and are simple and conducive to use in numerical procedures. In the following we propose criteria which satisfy these goals to various degrees.

An important feature of our intuition is that it tends to become more reliable as the data records increase in length, or more exactly, as the input data becomes more densely

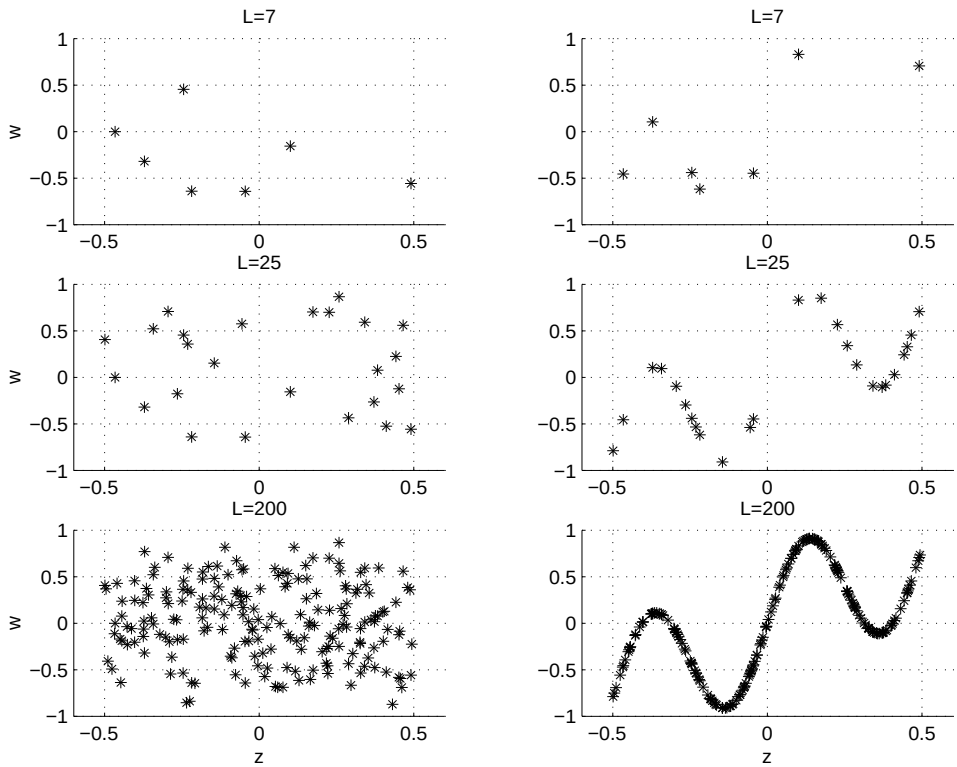


Figure 3.4: Intuition for associating a scatter plot with a static relation gets better as length of data record increases.

spaced. That is, the intuitive "smoothness" criteria becomes more robust at differentiating between input/output pairs that belong to static operators, and ones that don't. To put it another way, as the data length increases, the difference in smoothness grows more exaggerated between the scatter plot of signals with a dynamic relationship and that of signals with a static relationship.

To see what is meant consider another example where two systems (both SISO) are forced with identical input sequences. Scatter plots of the output versus the input are shown in Figure 3.4. The plots in the first column correspond to system 1 and those in the second column to system 2. The top row is scatter plots of the first seven input-output points in the data records, the middle row uses the first 25 pairs, and the lower row uses the first 200 pairs. If we were presented with the length-7 plots we would be hesitant to classify either system as static or not, and in fact it's difficult to distinguish any qualitative difference between the two. But if shown the length-25 plots we would probably strongly

suspect that system 1 is *not* static, especially if the data is generated by a system that we think is reasonably well behaved and not pathological in some way. At this point the scatter plot from system 2 still seems consistent with a static operator. Our early suspicions become stronger when confronted with the scatter plots of length 200. At this point it is easy to make the choice, for each plot, whether it is consistent with a static operator or not.

In view of this we should expect that our cost functions also become more reliable as the length of the data record increases. Consequently we will analyze the behavior of the proposed cost functions as a function of this length.

For a given system the data record obviously depends on what input is presented to it. A particular input may make for a more or less decisive scatter plot, and a bad choice may cause our intuition to fail even as the length of the observed data record increases. Thus there are some issues which are related to the persistence-of-excitation issues that come up in system identification and adaptive control. The intuition holds up well under the common situation that as the data length increases the points become more dense uniformly over the domain of interest.

In the previous section we saw that we can use the linear system constraints to parametrize the set of unknown signals which are feasible in terms of a free parameter f, and this will be our fundamental decision variable in some of the algorithms we propose. The cost functions that follow which judge the fitness of (z, w) as an input/output pair of $\mathcal{S}$ in turn judge the fitness of f by applying the criteria to (z_f, w_f).

Thus far the comments in this section apply to static operators with arbitrary numbers of inputs and outputs. As we formulate cost functions in the upcoming sections we will focus on single-output operators. This doesn't sacrifice generality. A general MIMO function $\phi(x)$ with m outputs and n inputs can be thought of as a collection of m single-output functions $\phi_i(x)$, of n inputs each:

$$\phi(x) = \begin{bmatrix} \phi_1(x) \\ \vdots \\ \phi_m(x) \end{bmatrix}.$$

Further, $\phi(\cdot)$ is static if and only if each of the $\phi_i(\cdot)$ is static. The cost functions to follow can be applied to each of the single-output ϕ_i, resulting in a problem with m cost functions; these can be individually constrained, or combined to form a scalar cost function as needed.

Before going farther we take a moment to introduce some notation. For a scalar

signal z we define P^z as the permutation operator which sorts the elements of z in increasing order. So for a scalar signal $w \in \mathbf{R}^L$, this operator rearranges the elements of w in a way that depends on z. As with all permutation operators, P^z is linear, and has a matrix representation, and we use P^z interchangeably to refer to the operator as well as its matrix representation. For example if $z = \begin{pmatrix} 3 & 2 & 1 & 4 \end{pmatrix}^T$, then we have

$$P^z = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Next, let $D^1 : \mathbf{R}^L \to \mathbf{R}^{L-1}$ be the first difference operator:

$$(D^1 x)(t) = x(t+1) - x(t),$$

and $D^2 : \mathbf{R}^L \to \mathbf{R}^{L-2}$ is the second difference operator:

$$(D^2 x)(t) = (D^1 x)(t+1) - (D^1 x)(t) = x(t+2) - 2x(t+1) - x(t).$$

These two operators take the differences between successive points of a vector, and are referred to as *neighboring-pairs* diff operators. Once again these are linear operators. For $L = 4$ we have the following matrix representations:

$$D^1 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \quad \text{and} \quad D^2 = \begin{bmatrix} 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \end{bmatrix}$$

We can also define an operator which computes the difference between *all* pairs. $D^{\text{ap}} : \mathbf{R}^L \to \mathbf{R}^{L(L-1)/2}$, and is defined so that

$$(D^{\text{ap}} x)(t) = \begin{pmatrix} x(2) - x(1) \\ x(3) - x(1) \\ \vdots \\ x(L) - x(1) \\ x(3) - x(2) \\ \vdots \\ x(L) - x(2) \\ \vdots \\ x(L) - x(L-1) \end{pmatrix}$$

This is referred to as the *all-pairs diff operator*. The associated matrix here is

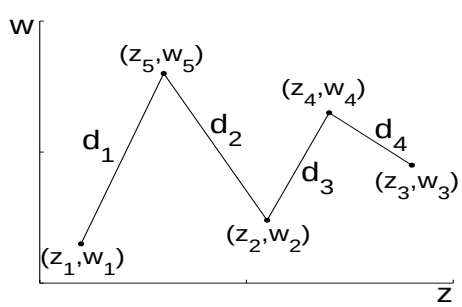
$$D^{\mathrm{ap}} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix},$$

For one of these matrices, such as D^1, D^1_k will refer to the k'th row of that matrix.

3.3.1 Dispersion Function

In this section we present a quantitative measure of the compatibility of an input-output pair (z, w) with a static relation between them. This measure does a good job of capturing staticness, and at the same time is somewhat amenable for use in convex programming solvers. First we consider SISO static operators, so the input and output signals z and w have size $n_z = n_w = 1$ and hence $(z(t), w(t)) \in \mathbf{R}^2$. Consider a cost function $\mathcal{J}(z, w)$ which for a particular data record (z, w) is the sum of the squares of the lengths in the $z - w$ plane of the lines connecting the points in the scatter plot, connecting the points in order of increasing z.

For instance consider the following scatter plot of a data record with 5 points, with the connecting lines drawn in



In this case the cost would be the sum of the squares of the four lengths d_i.

The quantity that is captured here is a notion of the "arclength" of the scatter plot. When $\{(z, w)\}$ is points lying on the graph of a static function, this measure is essentially a (non-uniform) discretization of the integral $\int dz^2 + dw^2 = \int ds^2$, where s is the arclength of the graph. Since $\int ds^2 = 0$ for continuous functions, we can expect this measure will

converge to zero as the density on the z-axis of the samples increases and the approximation of the discretization to the actual integral becomes better.

The measure matches our intuition for static operators in that it tends to be large for scatter plots whose points jump up and down erratically as you pass through them sequentially left to right, and it is smaller for scatter plots in which the points have a more orderly and monotonic pattern. In the first case the lengths between points is greater than when the scatter plot is more "smooth". This behavior is confirmed if we compute the cost function for the example of Figure 3.3; for the scatter plots shown there, $\mathcal{J}(z, w) = 4738$ for the dynamic system shown on the left side, while $\mathcal{J}(z, w) = 14$ for the static operator's scatter plot. This indicates that we may be able to rely on this function to distinguish between input/output data records based on to what degree they are consistent with a static relationship between them.

Notice that the order in which we connect the sample points for computing the cost function is generally different from their chronological order as they appear in the data record. Thus any algorithm computing the cost of a given data record (z, w) will at some point have to determine the ordering of the $z(t)$, i.e. *sort* them, in order to determine which distances are being squared and summed. This has important implications for how easy it is to use this function, as well as general properties the function possesses.

Using notation defined earlier we can make our definition of the cost function precise. Recall P^z is defined as the permutation operator which, when applied to the vector z, rearranges its elements in increasing order. For (z, w) a data record of length L, $(P^z z, P^z w)$ is the rearranged points so that they're sorted in time according to increasing values of z. Now the cost function described above, for such an input-output pair, is defined as

Definition 3.9 (dispersion) *For (z, w) a SISO data record of length L, the* dispersion *of that data record is defined to be*

$$\mathcal{J}(z, w) := \sum_{t=1}^{L-1} \left[(P^z z)(t+1) - (P^z z)(t)\right]^2 + \left[(P^z w)(t+1) - (P^z w)(t)\right]^2 \tag{3.102}$$

$$= \|D^1 P^z z\|_2^2 + \|D^1 P^z w\|_2^2. \tag{3.103}$$

We call this operator the *dispersion function*; it was previously introduced in [44] and [11]. The name reflects the intent of measuring how widely dispersed the points are from lying on the curve of a static function, or how cloudy the scatter plot looks.

Notice that the second term in the definition, as a function of w alone, is convex quadratic, which is attractive from an optimization standpoint. It can be handled in various solver software either as an objective function or a constraint. But as a function of both z and w it's not so nice, since P^z (the operator itself) depends on z in a complex, nonlinear, nondifferentiable way. Similarly the first term is convex quadratic in z only if we consider P^z to be fixed. This will be a factor in algorithms later on, for instance at times we will employ iterative schemes which successively fix the permutation operator, minimize over w, update the permutation, and repeat.

Remembering our setup where the estimates z_f and w_f are parametrized in terms of a free variable f, the dispersion as a function of f is

$$\mathcal{J}(f) := \|D^1 P^{z_f} z_f\|_2^2 + \|D^1 P^{z_f} w_f\|_2^2. \tag{3.104}$$

Recall from Section 3.2 the dependences of z_f and w_f on f are affine. In general these both depend on f, and so does the permutation operator P^{z_f}. Once again if the permutation operator is fixed then $\mathcal{J}(f)$ is convex quadratic in f.

The dispersion doesn't depend on the mean of the signals; we can add a constant to w and/or z and it doesn't change $\mathcal{J}(z, w)$:

$$\mathcal{J}(z + \alpha_1, w + \alpha_2) = \mathcal{J}(z, w).$$

That's reasonable. Scaling *does* change the dispersion:

$$\mathcal{J}(\alpha_1 z, \alpha_2 w) = \left\|D^1 P^z(\alpha_1 z)\right\|_2^2 + \left\|D^1 P^z(\alpha_2 w)\right\|_2^2 = \alpha_1^2 \left\|D^1 P^z z\right\|_2^2 + \alpha_2^2 \left\|D^1 P^z w\right\|_2^2.$$

This is somewhat undesirable. For instance such scalings may simply represent a change of units for the input and/or output of the operator in question. This clearly has nothing to do with whether or not that operator is static, yet it affects the dispersion measurement. Sometimes this means we need to choose weightings that account for the magnitude of the input and output of the operator, in order to normalize this effect. This is true when the system involves more than one static function to be identified, and the relative sizes of their dispersion functions is important. Choosing good weightings will often be a trial and error process. This is discussed more in Chapter 4.

We mentioned above that our intuition tends to get more reliable as the length of the data record increases. Let us investigate whether the dispersion function behaves

similarly. We'd like to convince ourselves that it can be used as a reliable indicator of staticness.

One thing to check is what happens in the limit as the data length L goes to infinity. When $\mathcal{S}$ is static, under a variety of conditions it can be shown the dispersion is bounded or goes to zero as $L \to \infty$. For example we can quickly show

Theorem 3.10 *Suppose $w = \mathcal{S}z$ with $\mathcal{S}$ a static map and total variation bounded by M. Then for any input record z (of any length L), which is contained within an interval $[a, b]$, $\mathcal{J}(z, w) \leq (b-a)^2 + M^2$.*

proof:

$$\begin{aligned}
\mathcal{J}(z,w) &= \sum_{t=1}^{L-1} \left(P_{t+1}^z z - P_t^z z\right)^2 + \sum_{t=1}^{L-1} \left(P_{t+1}^z w - P_t^z w\right)^2 \\
&\leq (b-a)^2 + \left(\sum_{t=1}^{L-1} \left|P_{t+1}^z w - P_t^z w\right|\right)^2 \\
&\leq (b-a)^2 + \left(\sup_{\zeta_1 = a \leq \zeta_2 \leq \cdots \leq \zeta_L = b} \sum_{i=1}^{L-1} \left|\mathcal{S}(\zeta_{i+1}) - \mathcal{S}(\zeta_i)\right|\right)^2 \\
&\leq (b-a)^2 + M^2
\end{aligned}$$

In the second-to-last line, the sup is over all size-L partitions of $[a, b]$, and this is bounded by the total variation, defined as the sup over all partitions of $[a, b]$. ■

Consequently, if the dispersion function is unbounded as L goes to infinity, assuming the input is bounded, then the operator is not a static operator with bounded variation. Or, consider stronger conditions on the input and $\mathcal{S}$, namely suppose $w = \mathcal{S}z$ with $\mathcal{S}$ a static, continuously differentiable map. Then it can be shown that as $L \to \infty$, if the the spacing of the input data z becomes uniformly increasingly dense, meaning the maximum gap between input points goes to zero, then the dispersion goes to zero [11]. This is expected because the integral $\int_I ds^2$ of the square arc length over some interval I of the graph of a continuous function is zero, and under these conditions the dispersion is a better and better approximation to this integral. Similar results are available in a stochastic setting. Suppose the data z is generated by an independent, identically distributed stochastic process, with each $z(t)$ having density function $p_Z(\cdot)$, which is bounded below by a constant $M > 0$ on some interval $[a, b]$. Then if we consider a dispersion criteria that only considers data points within that interval, the dispersion quantity converges to zero *in probability* [11].

These limit results are valuable to show that the dispersion criterion behaves in a reasonable fashion, that as the length of the data record increases the dispersion draws a sharper distinction between static and non-static maps. We expect that as we take advantage of more experiment data the we are better able to distinguish whether the data is consistent with a static relationship, and the above results help to confirm this.

Ultimately however data records are finite, and due to computational constraints sometimes the amount of data that can be handled is quite finite indeed. So it is also useful to look at finite data records, and how the behavior of the dispersion measure varies with data length. We'd like to get some kind of feel for how well the dispersion function separates static and dynamic systems. A convincing argument that it does a good job would involve analysis of its performance for various static and dynamic operators, as the input varies over a typical family of inputs that one might expect to see in practice.

One way to characterize the behavior of the dispersion function for a family of inputs is to consider z as a stochastic process. Then for a given operator $\mathcal{S}$ and data length L, the dispersion $\mathcal{J}(z, \mathcal{S}z)$ is a random variable with some probability density. In some sense this distribution function is a complete picture of the performance of the dispersion measure for that family of inputs. If the stochastic process characterizes a typical family of input signals that is seen in practice, then the distribution function describes the typical behavior of the dispersion measure. The hope is that in general the distributions associated with static $\mathcal{S}$ are small, centered near zero, and those for dynamic $\mathcal{S}$ are centered away from zero, and the two cases are well separated for large L. So that given a particular value of the dispersion associated with some signals (z, w), it is possible to classify it as coming from either a static or dynamic operator, with a high chance of being correct.

In general it's difficult to derive an analytical expression for the probability density function of the dispersion. This is true even if Z has a simple description (for instance independent identically-distributed (IID) Gaussian white noise) due to the complex nature of $\mathcal{J}(\cdot, \cdot)$. In some instances it's possible to derive expressions for the mean and variance for a given $\mathcal{S}$ as a function of L, but this is also usually hard.

By way of illustration, Monte Carlo simulation can be used to find the mean and variance numerically for specific $\mathcal{S}$ and distributions of the input.

Example 3.11 In order to define a realistic input process Z, start with an independent,

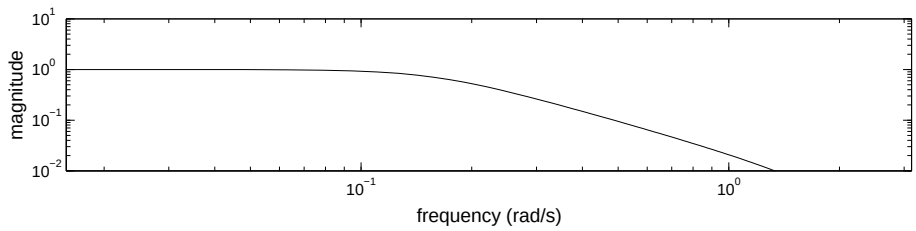


Figure 3.5: Frequency response of noise-coloring filter for input in Monte Carlo simulation of dispersion function.

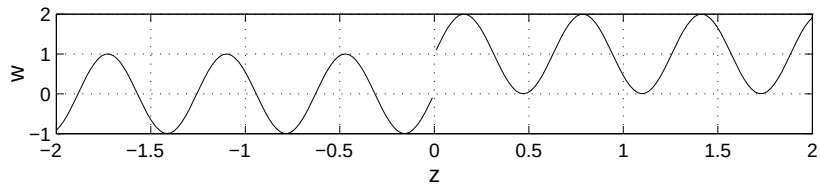


Figure 3.6: Operator $\mathcal{S}_2$.

identically distributed (IID) process U with $U(t)$ unit normally distributed,

$$U(t) \sim \mathcal{N}(0,1).$$

and define Z by

$$Z = \mathcal{L}U$$

where $\mathcal{L}$ is a linear filter. Thus Z is colored Gaussian noise. In this case $\mathcal{L}$ is a second-order butterworth filter with spectrum as in Figure 3.5. The coloring filter is introduced partly to make the example more realistic; in general the input to $\mathcal{S}$ is taken as the output of a dynamic system, and not controlled directly, and so there is correlation between the outputs at different times. We also want to illustrate the point that a given operator can be dynamic, yet look static when the input is slowly varying with respect to these dynamics. In practice the term "dynamic" is relative; it can be argued that almost *every* system is some dynamic, it's just that some operate so fast that from our point of view they are static. To make this point with the dispersion function we need an input that doesn't have arbitrarily high frequency components.

We will look the dispersion of four operators. $\mathcal{S}_1$ is the identity operator, clearly static. $\mathcal{S}_2$ is a more interesting static function, graphed in Figure 3.6.

Notice that it is discontinuous at zero, with a discontinuity of size one. We don't expect the dispersion of this operator to converge to zero as $L \to \infty$, but instead to a finite value. In this case we can reason that the finite value is unity. $\mathcal{S}_3$ and $\mathcal{S}_4$ are

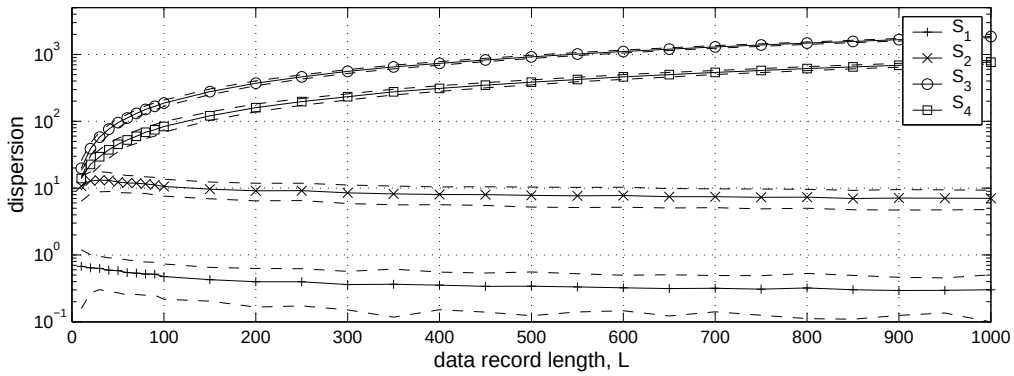


Figure 3.7: Sample means (solid lines) and ±1 sample standard deviation sleeve (dashed lines) of the dispersion $\mathcal{J}(z, \mathcal{S}_i z)$ with random inputs, for four operators.

dynamic operators. They are first-order linear systems:

$$\mathcal{S}_3 : \quad w_{k+1} = 0.9 w_k + 0.1 u_k \tag{3.105}$$

$$\mathcal{S}_4 : \quad w_{k+1} = 0.02 w_k + 0.98 u_k \tag{3.106}$$

Note the dynamics of $\mathcal{S}_4$ are considerably faster than those of $\mathcal{S}_3$. With respect to inputs that vary slowly, or at some bounded rate, or whose spectrum rolls off at high frequencies, $\mathcal{S}_3$ is "more dynamic" than $\mathcal{S}_4$; it responds to inputs more slowly.

In order to produce sample points of the dispersion, for a given L and $\mathcal{S}_i$ a realization of U is generated and filtered to compute z, then $w = \mathcal{S}_i z$ is computed. Before computing the dispersion of (z, w), we scale the size of w to have unit standard deviation. This makes the dispersion computations more comparable across the different operators $\mathcal{S}_i$. With respect to staticness, we should have no reason to prefer signal pairs that differ only by a scaling, yet the dispersion computation is affected by this, so we normalize the output w to have the same scaling.

Figure 3.7 shows results for several L between 10 and 1000. For each data record length, 500 samples of u are taken and the dispersion $\mathcal{J}(Lu, SLu)$ is computed for each. The sample mean of the dispersion is plotted, as well as a sleeve about the mean that is ±1 sample standard deviation. The important thing to note about this figure is not so much the absolute levels of the dispersion, but the trends as L increases, and the separation of the dispersion distributions for the static operators from those of the dynamic operators by several orders of magnitude. As L increases the dispersion for the dynamic operators grows quickly, while the dispersion of the static operators

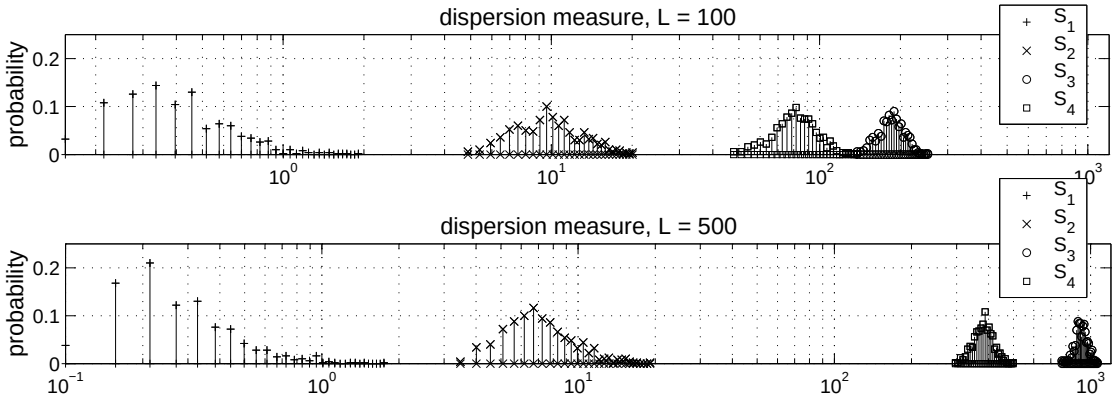


Figure 3.8: Empirical distributions of the dispersion measure for a random input, for two data lengths and four operators $\mathcal{S}_i$.

decreases. The sleeves show that the probability densities of the dispersion for the different operators are well-separated from each other.

Notice that the dispersion for the static $\mathcal{S}_2$ is larger than that of the static identity $\mathcal{S}_1$. This is primarily due to the discontinuity. As well, the dispersions of the dynamic systems differ, that of $\mathcal{S}_4$ is always smaller than that of $\mathcal{S}_3$. This is consistent with the idea that $\mathcal{S}_3$ has a faster response than $\mathcal{S}_4$ and so with respect to the frequency-limited input signal it appears more static. Its dispersion is still much larger than either of the two truly static systems, however.

We can also look at histograms of samples of the dispersion, which form an approximation of the probability density function of the dispersion measure (the *empirical histogram*) [34]. In Figure 3.8 we chose to look at 2000-sample-point histograms for two data record lengths, $L = 100$ and $L = 500$. These are computed using 30 equally-spaced bins. Once again we see that for the larger data lengths, the distributions of the static and dynamic operators are well separated. □

Similar results are obtained for other classes of input. In general the division between static and dynamic systems is more striking when the input is white noise, and the $z(t)$ for different times are independent. More deterministic-looking input signals, such as sine sweeps, can also do a good job. For instance, Figure 3.9 is the analog of Figure 3.7 where now the input is deterministic instead of colored noise; a sine chirp, linear in frequency:

$$z(t) = 2\sin\left[2\pi\left(0.025 + (0.1 - 0.025)\frac{t}{L}\right)t\right].$$

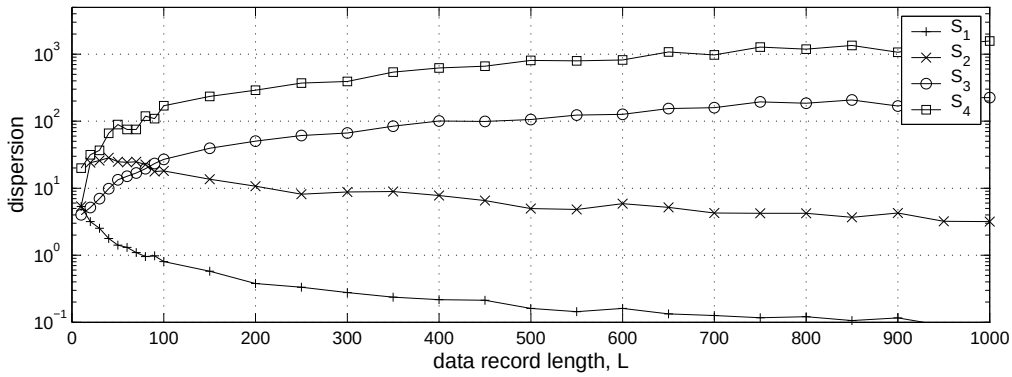


Figure 3.9: Dispersion $\mathcal{J}(z, \mathcal{S}_i z)$ with sine chirp input, for four operators.

Two properties that make a signal a good candidate for separating static and dynamic systems are

1. the points $z(t)$ form a dense covering of some part of the domain of $\mathcal{S}$; this will cause the dispersion of static systems to be small, and
2. the input contains frequency components that are high enough to reveal the dynamics of systems that those systems we want to consider as dynamic, so that the dispersion of these systems is large.

Realizations of white and colored noise tend to do well at both. Criteria such as these can aid in designing experiments that lead to good estimates. There is related material regarding what makes input signals favorable in Chapter 5.

At any rate these explorations of how the dispersion function behaves for various input signals and static and dynamic operators make a convincing argument that the dispersion function is a reasonable measure of staticness. If we are looking for input-output estimates that are consistent with a static relationship then their dispersion should be small, because the dispersion of a static function is reliably small, and that of a dynamic function is reliably large and grows with data record length. This holds for a wide array of different operators and input signals (white noise realizations, colored noise realizations, sufficiently exciting deterministic signals), and it's fairly clear what properties make an input a good choice for differentiating between static and non-static operators. For these inputs, the distinction drawn between the two types of operators grows more clear as the data record length becomes larger.

This motivates the idea of minimizing dispersion for candidate estimates of z and

w. Whether or not actually minimizing is the correct thing to do might be questionable, because for any finite L and all but degenerate inputs, the dispersion will achieve some value greater than zero, even for static operators. In reality we will see that this generally doesn't present a problem. In simulation examples the dispersion of the actual input-output signals is usually lower than the estimated signals. The achieved dispersion is usually very small (and once again this is more and more true as L increases), and the minimization problems generally don't over-optimize to achieve a smaller value. Nevertheless it may be valuable to keep this issue in mind. Another option is to *constrain* the dispersion measure of estimated signals, while minimizing some other property of interest such as another staticness or smoothness measure, or some estimate of a stochastic property of the candidate signals.

3.3.2 Related Staticness/Smoothness Measures

arclength

Looking back at Section 3.3.1 the dispersion is defined as the sum of the squares of the lengths of the connecting lines in the scatter plot. A similar quantity is the arclength of the scatter plot, the sum of the *lengths* of the connecting lines:

$$\mathcal{J}_{\mathrm{A}}(z,w) := \sum_{t=1}^{L-1} \left((D_t^1 P^z z)^2 + (D_t^1 P^z w)^2 \right)^{1/2}. \tag{3.107}$$

Recall for continuous static operators the dispersion measure goes to zero as the density of z increases. In contrast the limiting value of $\mathcal{J}_{\mathrm{A}}$ is the arclength of the graph of the function. $\mathcal{J}_{\mathrm{A}}$ is a nonconvex nondifferentiable function of z and w due to the dependence of P^z on z. When P^z is replaced by a fixed matrix independent of z then it is differentiable, and it is possible to use $\mathcal{J}_{\mathrm{A}}$ as an objective or constraint in convex programs by introducing auxiliary variable into the problem.

total variation

If in the arclength summation we drop the part of the length in the z direction and sum just the length in the w direction, we are left with the *total variation* of the (linear interpolation of the) scatter plot:

$$\mathcal{J}_{\mathrm{TV}}(z,w) := \sum_{t=1}^{L-1} \left| D_t^1 P^z w \right| = \| D^1 P^z w \|_1 \tag{3.108}$$

The same comments apply as for arclength regarding convexity and behavior as z becomes more dense.

neglecting the z part of $\mathcal{J}$

At times a simplified version of the dispersion function consisting of only the second term in (3.103) is useful:

$$\mathcal{J}_w(z, w) := \|D^1 P^z w\|_2^2 \tag{3.109}$$

or

$$\mathcal{J}_w(f) = \left\|D^1 P^{z_f} w_f\right\|_2^2 = \left\|D^1 P^{z_f}(w^0 + K_w f)\right\|_2^2. \tag{3.110}$$

$\mathcal{J}_w$ has a similar relationship to the full dispersion $\mathcal{J}$ as the total variation measure $\mathcal{J}_{\mathrm{TV}}$ has to the arclength $\mathcal{J}_{\mathrm{A}}$. If all elements of z are measured or assumed known then the first term is fixed and known and has no consequence for estimating unknown signals, leaving only $\mathcal{J}_w$. Even if the first term is not fixed, it is small for inputs z such that the sorted points $P^z z$ are closely spaced, and it can still be sensible to neglect it. Note this holds regardless of the operator $\mathcal{S}$ since nothing in the first term depends on $\mathcal{S}$ (it depends only on z). So at times we will use the simplified dispersion function (3.109) even when z is not known. That is we neglect the contribution of the separation of the inputs to the cost function, and only look at the output values. In our experience, the intuitive notion that the scatter plot should be orderly is preserved under this modification.

second-difference dispersion

A variation on the dispersion is to use a higher-order difference than in the definition of $\mathcal{J}$. For instance we can define a dispersion that uses second differences:

$$\mathcal{J}_2(z, w) := \sum_{t=1}^{L-2} \left[P_{t+2}^z z - 2P_{t+1}^z z + P_t^z z\right]^2 + \left[P_{t+2}^z w - 2P_{t+1}^z w + P_t^z w\right]^2 \tag{3.111}$$

$$= \|D^2 P^z z\|_2^2 + \|D^2 P^z w\|_2^2, \tag{3.112}$$

and dropping the z part we have

$$\mathcal{J}_{2w}(z, w) := \sum_{t=1}^{L-2} \left[P_{t+2}^z w - 2P_{t+1}^z w + P_t^z w\right]^2. \tag{3.113}$$

Whereas $\mathcal{J}$ penalizes the sum of squares of distances between points in the scatter plot, $\mathcal{J}_2$ penalizes the sum of squares of the *difference* in distance between successive pairs of points,

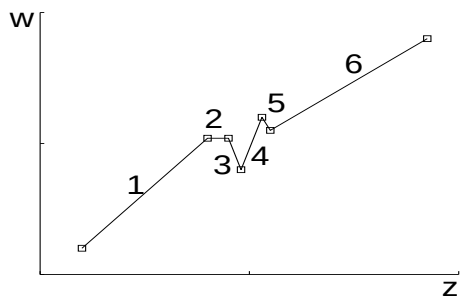


Figure 3.10: Scatter plot with outlier points.

a different type of smoothness. Using both $\mathcal{J}$ and $\mathcal{J}_2$ in a problem allows more control over the nature of the scatter plot of the resulting estimates.

sum of slopes (first and second derivatives)

One feature of the dispersion function that is perhaps undesirable occurs when there are relatively large gaps in the input data. There may be two input-output pairs, $(z(t), w(t))$ and $(z(s), w(s))$, with $z(t)$ and $z(s)$ spaced far apart with no input pairs occurring between them on the z axis. For instance this happens when the input is normally distributed white noise, which will contain a small number of points in the tails of the distribution. Even when the operator is static the $w(t), w(s)$ might then be far apart as well. In this case the term $(|z(t) - z(s)|^2 + |w(t) - w(s)|^2)$ makes a large contribution to the dispersion, even though this situation is not incompatible with a static, or even smooth, operator. Instead it's an artifact of an unfortunate sparsely-spaced data pair, and the squaring nature of the dispersion function. But the procedure for forming estimates of z and w will tend to want to produce a pair that is closer together, to minimize this cost. For z's that are spaced far apart, in some sense we would like to allow the difference in the w's to be large as well, and to penalize that less.

Indeed it is often the case that the input data will be sparsely spaced near the edges of the interval in which the data is contained, and there will be a few outliers. This motivates us to consider other measures of staticness that can be used instead of dispersion, or, more generally, to augment the dispersion measure. Consider the 7 point scatter plot of Figure 3.10. Here there are 5 points spaced relatively closely, as compared to the outlying points on either side. These two points are penalized substantially more than the rest by

the dispersion.

One idea is to penalize scatter plots which involve large derivatives. Define $\mathcal{J}_{\mathrm{D1}}(z,w)$ as the sum of the squares of the slopes of the lines drawn between successive points in the scatter plot. By penalizing the approximate first derivative, we discourage oscillations and noise-like behavior in the scatterplot. In the scatter plot of Figure 3.10 there are six terms in $\mathcal{J}_{\mathrm{D1}}$. The slopes of lines 3, 4, and 5 are larger than those of 1 and 6, and so the sort of up-and-down jiggling of the middle points incurs a greater cost than the derivatives associated with the two extreme points. This is a smoothness penalty. We approximate the derivative of the scatter plot with first-order differences (using notation defined in earlier):

$$\partial_t^1(w,z) := \frac{D_t^1 P^z w}{D_t^1 P^z z}$$

and sum the squares to produce this cost function,

$$\mathcal{J}_{\mathrm{D1}}(z,w) := \sum_{t=1}^{L-1} \left(\partial_t^1(w,z)\right)^2 \tag{3.114}$$

$$= \sum_{t=1}^{L-1} \left(\frac{D_t^1 P^z w}{D_t^1 P^z z}\right)^2 \tag{3.115}$$

$$= \left\| D^1 P^z w \,./\, D^1 P^z z \right\|_2^2 . \tag{3.116}$$

(Here, given two vectors of the same length we use $'./'$ to mean element-by-element division, producing another vector of the same length: $(a./b)(t) := a(t)/b(t)$.)

Going further, we could also approximate second derivatives by differencing. Define the operator

$$\begin{aligned}
\partial_t^2(w,z) &:= \frac{\partial_{t+1}^1(w,z) - \partial_t^1(w,z)}{\left(\frac{z_{t+2}+z_{t+1}}{2} - \frac{z_{t+1}+z_t}{2}\right)} \\
&= \frac{2}{(z_{t+2}-z_t)}\left[\frac{w_{t+2}-w_{t+1}}{z_{t+2}-z_{t+1}} - \frac{w_{t+1}-w_t}{z_{t+1}-z_t}\right] \\
&= 2\left[\frac{w_{t+2}-w_{t+1}}{(z_{t+2}-z_t)(z_{t+2}-z_{t+1})} - \frac{w_{t+1}-w_t}{(z_{t+2}-z_t)(z_{t+1}-z_t)}\right] \\
&= 2\left[\frac{w_t}{(z_{t+2}-z_t)(z_{t+1}-z_t)} - \frac{w_{t+1}}{(z_{t+2}-z_{t+1})(z_{t+1}-z_t)} + \frac{w_{t+2}}{(z_{t+2}-z_t)(z_{t+2}-z_{t+1})}\right]
\end{aligned}$$

This leads to a second derivative cost

$$\mathcal{J}_{\mathrm{D2}}(z,w) := \sum_{t=1}^{L-1} \left(\partial_t^2(P^z w, P^z z)\right)^2 \tag{3.117}$$

Instead of large slopes, $\mathcal{J}_{\mathrm{D2}}$ penalizes large changes in slope.

First and second derivative criteria and combinations of them, along with the Lipschitz bounds to be discussed in Section 3.3.4, are more traditional measures of smoothness of a function. As usual $\mathcal{J}_{\mathrm{D1}}$ and $\mathcal{J}_{\mathrm{D2}}$ are nice convex functions of w but messy functions of z. As well, they can be sensitive to z, in that small perturbations from a nominal z can result in large changes in their value. This can make them harder to use in practice, especially in problems where z is not known and needs to be estimated. For instance this applies to the "bootstrapping" ideas for solving these problems that are introduced in Chapter 4, where intermediate approximations to z are used in successive iterations. We have found the dispersion measures $\mathcal{J}$ and $\mathcal{J}_2$ are much less sensitive to perturbations in z while still capturing the essential behavior of $\mathcal{J}_{\mathrm{D1}}$ and $\mathcal{J}_{\mathrm{D2}}$, so having introduced the latter, in practice we will stick to using the dispersion.

comparisons

Depending on prior knowledge about the candidate $\mathcal{S}$ being identified there may be reason, in enforcing staticness or smoothness of estimated nonlinearities, to prefer to use one of $\mathcal{J}$ ($\mathcal{J}_w$), $\mathcal{J}_2$ ($\mathcal{J}_{2w}$), $\mathcal{J}_{\mathrm{A}}$ ($\mathcal{J}_{\mathrm{TV}}$), $\mathcal{J}_{\mathrm{D1}}$, $\mathcal{J}_{\mathrm{D2}}$, or Lipschitz constraints (to be discussed in Section 3.3.4). Or a certain combination of these.

In the same way that Example 3.11 looked at the behavior of the dispersion function as the data record length grows, for the operators $\mathcal{S}_k$, $k = 1..4$ and using colored noise input, we can also look at these other dispersion-like measures. Analogously to Figure 3.7, Figure 3.11 shows the sample means for seven of the staticness measures, for the four operators and the same input data as previously.

The data for $\mathcal{J}$ is the same as in Figure 3.7. The modified dispersion function $\mathcal{J}_w$ is also shown; notice that its difference from the full $\mathcal{J}$ is small. The same comment applies for $\mathcal{J}_{\mathrm{A}}$ and $\mathcal{J}_{\mathrm{TV}}$; they are nearly equal. All of the criteria have the property of being much larger for the two dynamic operators than for the two static ones. For the dynamic operators, each measure increases with increasing data record length (increasing z density). For the static operators, the behavior as L increases differs. With $\mathcal{J}$, $\mathcal{J}_w$, and $\mathcal{J}_2$ the value decreases as density increases, for $\mathcal{J}_{\mathrm{A}}$ and $\mathcal{J}_{\mathrm{TV}}$ the value approaches a constant (the arclength or total variation respectively), and for $\mathcal{J}_{\mathrm{D1}}$ and $\mathcal{J}_{\mathrm{D2}}$ the value increases (but is still much smaller than for either of the dynamic operators).

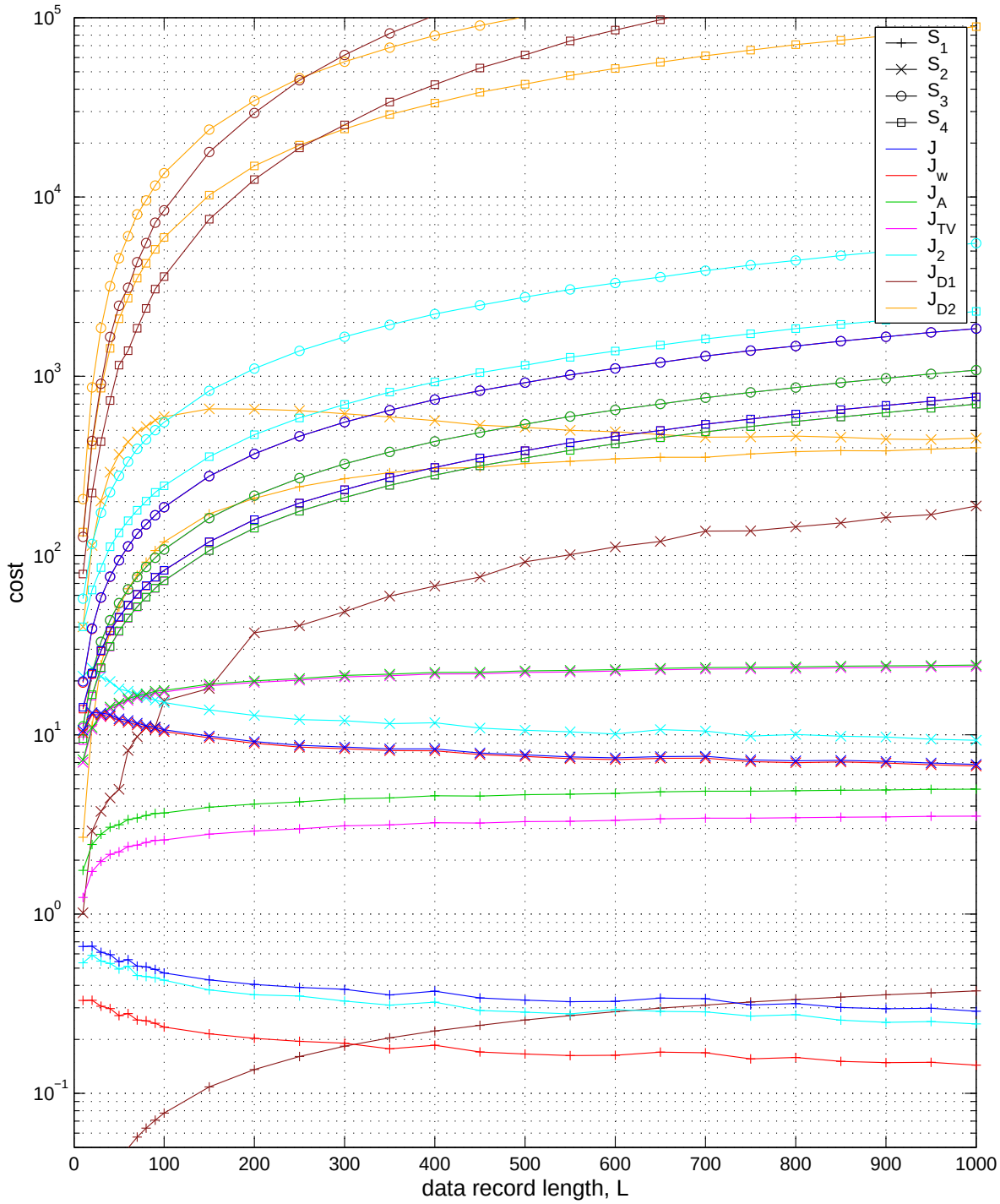


Figure 3.11: Sample means (random inputs) vs record length L for 8 of the dispersion-like measures ($\mathcal{J}$, $\mathcal{J}_w$, $\mathcal{J}_{\mathrm{A}}$, $\mathcal{J}_{\mathrm{TV}}$, $\mathcal{J}_2$, $\mathcal{J}_{\mathrm{D1}}$, and $\mathcal{J}_{\mathrm{D2}}$), for four operators each. In this figure $\mathcal{J}_{\mathrm{D1}}$ and $\mathcal{J}_{\mathrm{D2}}$ are divided by 1000 and 100000, respectively.

3.3.3 Extensions of Dispersion to Multi-Input Operators

The Lipschitz conditions discussed in Section 3.3.4 apply equally well to multi-input $\mathcal{S}$ as well as to scalar-input operators (although in some situations where $\mathcal{S}$ is MIMO we may have reason to consider a more structured set of Lipschitz constraints—see Section 3.3.5). But the dispersion function as defined in Section 3.3.1 applies only to single-input $\mathcal{S}$ only. It can however be extended to multi-input operators in a relatively straightforward manner, while retaining many of the same features (both desirable and undesirable) of the scalar case.

To show the way to do this we consider now a *two*-input, single-output mapping. Analogous to the arclength-like measure of Section 3.3.1, we extend the definition of the dispersion function to be an *area*-like measure of the two-dimensional manifold defined by the graph of this function. The main issue introduced by the extra dimension is how, given a finite set of points (z, w) in three-dimensional space supposedly lying on the graph of the function, to connect them to form a two-dimensional object that approximates the graph, and what an appropriate measure of the dispersion of this object would be.

In the two-input case the input vector z consists of points $z(t) = (z_1(t), z_2(t)),\ t = 1..L$ lying in the z_1–z_2 plane. One way to proceed is to use a triangulation of these points to in turn define an interpolation of the data (z, w) which is a two-dimensional surface. The three points for the k'th triangle are indexed by the three time points $(t_{k,1}, t_{k,2}, t_{k,3})$:

$$z(t_{k,1}) = \begin{bmatrix} z_1(t_{k,1}) \\ z_2(t_{k,1}) \end{bmatrix}, \quad z(t_{k,2}) = \begin{bmatrix} z_1(t_{k,2}) \\ z_2(t_{k,2}) \end{bmatrix}, \quad z(t_{k,3}) = \begin{bmatrix} z_1(t_{k,3}) \\ z_2(t_{k,3}) \end{bmatrix}, \quad k = 1..M. \tag{3.118}$$

The number M of triangles formed in a given triangulation procedure depends on the particular algorithm used, the length L of the data, and the actual value of z. A good candidate is Delaunay triangulation, which is popular due to desirable geometric properties. In this case the number of triangles produced is $O(L)$, where L is the number of points in the set, in this case equal to the data length. Empirically, with points selected randomly, we find the number is roughly $2L$. Now tack on the corresponding w points and consider the convex hull of each triplet of points, which defines M facets:

$$F_k := \mathbf{Co}\left(\begin{bmatrix} z(t_{k,1}) \\ w(t_{k,1}) \end{bmatrix}, \begin{bmatrix} z(t_{k,2}) \\ w(t_{k,2}) \end{bmatrix}, \begin{bmatrix} z(t_{k,3}) \\ w(t_{k,3}) \end{bmatrix} \right), \quad k = 1..M. \tag{3.119}$$

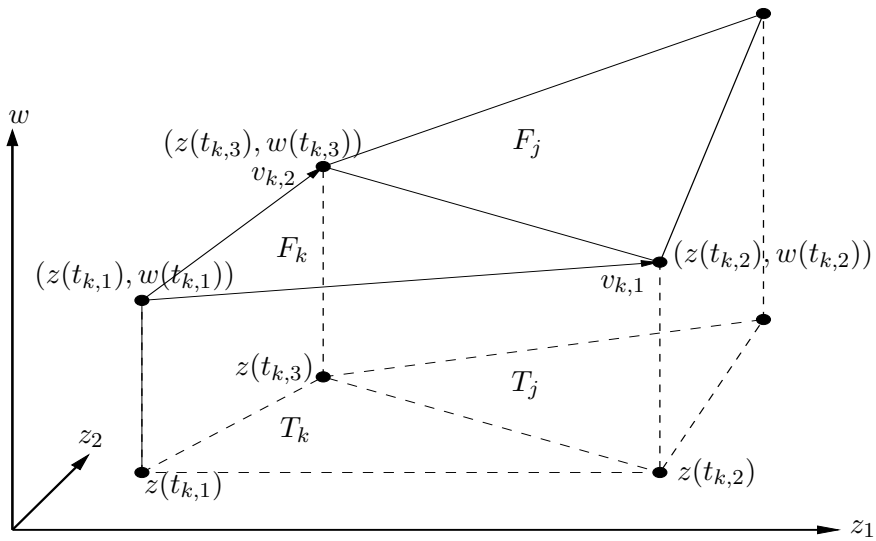


Figure 3.12: Construction of two-input dispersion function.

The union of these facets defines a surface that is a reasonable way to interpolate the data points to produce an estimate of the function graph they might belong to.

Analogously to Section 3.3.1, where we defined the dispersion of single-input data to be the sum of the squares of the lengths of the interpolating line segments of the sorted data, here we define the dispersion of two-input data to the the sum of the squares of the areas of the interpolating facets of the triangulated data:

Definition 3.12 (dispersion, two-input) *For (z, w) a single output, two-input data record of length L, the dispersion of that data record is defined to be*

$$\mathcal{J}(z, w) := \sum_{k=1}^{M} \text{area}^2(F_k) \tag{3.120}$$

When the data is a set of points lying on the graph of a surface, this sum approximates the integral of the squared-area over a portion of the surface, $\int dA^2$.

As for actual computation, recall that use the norm of the cross-product between any two sides of a facet is equal to the area of its facet. Consider Figure 3.12, where a portion of the triangulation of z and the resulting triangles and facets are shown. Consider facet F_k. Choosing two of its sides $v_{k,1}$ and $v_{k,2}$ as shown, the area of the facet can be

computed as

$$\begin{aligned}
\text{area}^2(F_k) &= \|v_{k,1} \times v_{k,2}\|^2 && (3.121)\\
&= \left\| \begin{bmatrix} z_1(t_{k,2}) - z_1(t_{k,1}) \\ z_2(t_{k,2}) - z_2(t_{k,1}) \\ w(t_{k,2}) - w(t_{k,1}) \end{bmatrix} \times \begin{bmatrix} z_1(t_{k,3}) - z_1(t_{k,1}) \\ z_2(t_{k,3}) - z_2(t_{k,1}) \\ w(t_{k,3}) - w(t_{k,1}) \end{bmatrix} \right\|^2 && (3.122)\\
&= \left\| \begin{bmatrix} (z_2(t_{k,2}) - z_2(t_{k,1}))\,(w(t_{k,3}) - w(t_{k,1})) - (z_2(t_{k,3}) - z_2(t_{k,1}))\,(w(t_{k,2}) - w(t_{k,1})) \\ -(z_1(t_{k,2}) - z_1(t_{k,1}))\,(w(t_{k,3}) - w(t_{k,1})) + (z_1(t_{k,3}) - z_1(t_{k,1}))\,(w(t_{k,2}) - w(t_{k,1})) \\ (z_1(t_{k,2}) - z_1(t_{k,1}))\,(z_2(t_{k,3}) - z_2(t_{k,1})) - (z_2(t_{k,2}) - z_2(t_{k,1}))\,(z_1(t_{k,3}) - z_1(t_{k,1})) \end{bmatrix} \right\|^2 . && (3.123)
\end{aligned}$$

As a function of (z, w), the two-input dispersion (3.120) is a messy function, composed of the triangulation step, followed by summing the squares of the facets. As a function of w alone however it is convex quadratic. To see this rewrite (3.123) as

$$\text{area}^2(F_k) = \left\| b + \begin{bmatrix} z_2(t_{k,3}) - z_2(t_{k,2}) & z_2(t_{k,1}) - z_2(t_{k,3}) & z_2(t_{k,2}) - z_2(t_{k,1}) \\ z_1(t_{k,2}) - z_1(t_{k,3}) & z_1(t_{k,3}) - z_1(t_{k,1}) & z_1(t_{k,1}) - z_1(t_{k,2}) \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w(t_{k,1}) \\ w(t_{k,2}) \\ w(t_{k,3}) \end{bmatrix} \right\|^2 \tag{3.124}$$

with

$$b = \begin{bmatrix} 0 \\ 0 \\ (z_1(t_{k,2}) - z_1(t_{k,1}))\,(z_2(t_{k,3}) - z_2(t_{k,1})) - (z_2(t_{k,2}) - z_2(t_{k,1}))\,(z_1(t_{k,3}) - z_1(t_{k,1})) \end{bmatrix} .$$

Hence the squared area of an individual facet is expressed as the squared norm of a linear function of the elements of w. This is a similar situation to that which we saw in the single-input case, where as a function of w and z the dispersion is messy, but as a function of w along it is convex quadratic.

The behavior of the two-input dispersion function for various static and dynamic operators, as a function of length, has similar features to the single-input version, as discussed in Section 3.3.1. In particular its value tends to decrease with increasing data length for static operators, while it increases for dynamic operators. For static operators of bounded variation, the dispersion can be bounded above; as data becomes increasingly dense the dispersion approaches a finite value, which is zero if the operator is continuous. These results, and results for stochastic settings, can be found in [11].

The procedure can be extended to operators with more than two inputs. Generalizations of Delaunay triangulation exist in three and higher dimensions. For instance, three-dimensional space gets partitioned into tetrahedrons, which are the analogs of triangles in two-dimensional space. In general one can define the dispersion as the sum of the "areas" of the convex hulls of the points associated with the "hyper-triangles".

3.3.4 Lipschitz Smoothness

A common notion of the smoothness of a function is Lipschitz continuity. A function $f : D \to R$ is said to be Lipschitz continuous if there exists $\gamma \in \mathbf{R}$ such that

$$\|f(x) - f(y)\|_R \leq \gamma \|x - y\|_D \quad \forall x, y \in D. \tag{3.125}$$

In this case we say f has Lipschitz bound γ, and we write $f \in L_\gamma$ to denote this situation. This is a essentially a bound on the derivative of f. The definition (3.125) applies to functions with an arbitrary number of inputs and outputs, using appropriate choices of the norms.

Prior knowledge about $\mathcal{S}$, or some part of $\mathcal{S}$, may include Lipschitz bounds. Whereas the staticness criteria that are mentioned above are concerned with an integral of a quantity like arclength over the domain of $\mathcal{S}$, or a sum of the slopes of the scatter plot, a Lipschitz constraint is a pointwise bound on the derivative. For instance this can be used to provide finer control over more localized smoothness properties of estimates.

The finite-length discrete sequence (z, w) is a possible input-output pair of an operator with Lipschitz bound γ if and only if the points in the scatter plot satisfy (3.125) pairwise. Using the 2-norm, we have the resulting set of constraints

$$\|w(t) - w(s)\|_2 \leq \gamma \|z(t) - z(s)\|_2 \qquad \forall t, s \in T. \tag{3.126}$$

With a more general weighted 2-norm allows we can adjust the smoothness for each input direction independently, e.g.

$$\|w(t) - w(s)\|_2^2 \leq (z(t) - z(s))^T G^T G (z(t) - z(s)) \qquad \forall t, s \in T, \tag{3.127}$$

where G is a weighting matrix.

For a data record of length L, (3.126) or (3.127) is $L(L-1)/2$ inequalities. However if the relation is single-input, it's equivalent to check just the adjacent pairs (triangle

inequality), and the pair (z, w) is consistent with Lipschitz constant γ if and only if

$$\|P^z_{t+1}w - P^z_t w\|_2 \leq \gamma |P^z_{t+1}z - P^z_t z| = \gamma |D^1_t P^z z|, \quad t = 1..L-1. \tag{3.128}$$

This is $O(L)$ constraint functions, whereas (3.126) is $O(L^2)$. In order to use this reduced set of conditions it is necessary to know the ordering of z, i.e. P^z.

How well suited are these constraints for various optimization frameworks? The constraints of (3.126), written in a more canonical form, are

$$\|w(t) - w(s)\|_2 - \gamma \|z(t) - z(s)\|_2 \ \leq 0, \qquad \forall t, s \in T \tag{3.129}$$

The left hand side here is function that is quadratic, and convex in w but nonconvex in z, and so in general (3.126) is a set of constraints that is quadratic but nonconvex in the decision variables. Similarly, the constraints (3.128) are convex quadratic in w, but nonconvex nonquadratic in z. If P^z is replaced with a fixed permutation operator then it becomes quadratic, but still nonconvex, in z.

An obvious special case is when z is fixed. In this case (3.126) and (3.128) are sets of $L(L-1)/2$, resp. $(L-1)$ convex quadratic constraints in w—a large number, but the constraints are attractive for optimization. If, further, w is scalar, then these can be replaced with *linear* inequality constraints in w (even better); (3.126) becomes

$$(w(t) - w(s)) \leq \gamma \|z(t) - z(s)\|_2, \qquad \forall t, s \in T, \tag{3.130}$$

$$-(w(t) - w(s)) \leq \gamma \|z(t) - z(s)\|_2, \qquad \forall t, s \in T, \tag{3.131}$$

$L(L-1)$ linear inequality constraints in w, and (3.128) becomes

$$(P^z_{t+1}w - P^z_t w) \leq |P^z_{t+1}z - P^z_t z|, \quad t = 1..L-1, \tag{3.132}$$

$$-(P^z_{t+1}w - P^z_t w) \leq |P^z_{t+1}z - P^z_t z|, \quad t = 1..L-1, \tag{3.133}$$

$2(L-1)$ linear inequality constraints in w.

In some situations it is possible to use a Lipschitz bound that varies over the domain of $\mathcal{S}$. For instance this can be useful when $\mathcal{S}$ is known to have a sharp transition or discontinuity somewhere in an uncertain interval, and otherwise its derivative is small. A z-dependent Lipschitz bound can be used to take advantage of an *a priori* picture of the general shape of $\mathcal{S}$. As always the model set, the set of unknown signals, and solution framework being used in a particular situation determine whether it is possible to use such information.

3.3.5 Other Structure in $\mathcal{S}$

Multi-Output

Up to this point we have concentrated on developing staticness criteria and constraints for single-output $\mathcal{S}$ (Section 3.3.4 is the exception that can be applied without change for a general number of outputs). However in general the model structure described in Section 2.3 allows for $\mathcal{S}$ that are collections of several single-output static functions. We can formulate a measure of staticness between the output and inputs, or indeed several measures of staticness and/or smoothness, for each of these. Most generally we would choose to constrain some of these measures, leaving the rest to be minimized.

This means one is potentially left with a multi-objective optimization problem. The most common way to proceed is to create a single-objective problem by using a weighted sum of the individual criteria. For instance, if $\mathcal{S}$ consists of n_w SISO functions $\mathcal{S}_k$, if we weight the k'th function by Γ_k then the dispersion function would be

$$\sum_{k=1}^{n_w} \Gamma_k \Big(\|D^1 P^{z_k} z_k\|_2^2 + \|D^1 P^{z_k} w_k\|_2^2 \Big). \tag{3.134}$$

Formulation of optimization problems to generate estimates now involves choosing values for the design variables Γ. The choice should be guided by the relative sizes of the component functions (remember dispersion is not scale-invariant) and the relative sizes of the contributions of the component functions in the output y. The weights can also affect how accurate the component estimates are relative to one another. Ultimately, experience and trial-and-error is important for choosing good values.

The second thing to point out here is that in general it is natural to model each output of $\mathcal{S}$ as being a function of only a subset of the entire set of inputs to $\mathcal{S}$. For instance in Example 2.1 there were three functions to identify, two spring force characteristics, $k_1(x_1 - x_2)$ and $k_2(x_2)$, and a friction force $f_x(\dot{x}_2)$. $\mathcal{S}$, being the union of these functions, has inputs $x_1 - x_2$, x_2, and $\dot{x}_2$, but the first output depends only on $x_1 - x_2$, the second only on x_2, and the third only on $\dot{x}_2$. These are modeling considerations, based on *a priori* knowledge about the system under investigation.

It is easy to take this type of model structure into account. Simply formulate the staticness measures to include the relevant inputs only. Given vectors z_1, z_2, and w, it is a stronger statement to say that the relationship from z_1 to w is static, than to say the relationship from (z_1, z_2) to w is static. For one, if z_1 has repeated values, it is

possible that (z_1, z_2) does not (recall the discussion at the beginning of Section 3.3). If *a priori* considerations dictate that an output does not depend on some signals, using this information leads to more effective staticness measures, and presumably then, better signal estimates.

Similar ideas hold for smoothness considerations. In Section 3.3.4 we showed how a Lipschitz condition on $\mathcal{S}$ translates into the set of constraints (3.126), one for each pair of input points in the scatter plot. When $\mathcal{S}$ is multi-output, composed of the n single-output functions $\mathcal{S}_1(\cdot), \ldots, \mathcal{S}_n(\cdot)$, we can alternatively prescribe a different Lipschitz constant for each output:

$$|\mathcal{S}_k(x) - \mathcal{S}_k(y)| \leq \gamma_k \|x - y\|_D \quad k = 1..n, \ \forall x, y \tag{3.135}$$

or in terms of z and w,

$$|w_k(t) - w_k(s)| \leq \gamma_k \|z(t) - z(s)\|_2 \quad k = 1..n, \ \forall s, t \in T. \tag{3.136}$$

When an output depends on only a subset of the inputs we again modify the constraint accordingly. The Lipschitz condition becomes

$$|w_k(t) - w_k(s)| \leq \gamma_k \|z_{[k]}(t) - z_{[k]}(s)\|_2 \quad k = 1..n, \ \forall s, t \in T, \tag{3.137}$$

where we use the notation $z_{[k]}$ to mean the subset of the n_z scalar signals comprising z which the k'th output depends on. This is a stronger, more specialized condition, than (3.136). When $\mathcal{S}$ is "diagonal", that is a collection of n SISO functions where the k'th output depends on only the k'th input, this set of Lipschitz conditions becomes

$$|w_k(t) - w_k(s)| \leq \gamma_k |z_k(t) - z_k(s)| \quad k = 1..n, \ \forall s, t \in T. \tag{3.138}$$

Repeated Elements

Sometimes there is reason to believe that in a given system, the same static function shows up in more than one place, even though the function itself is not known. For instance a device may contain multiple copies of essentially identical parts, manufactured in the same way. Or the same fundamental physical phenomenon acts in multiple places. This leads to model sets with *repeated* unknown elements. In this case it is desirable to both constrain estimates of such elements to be identical, and also to take advantage of this extra knowledge during identification.

This is easy to handle in the current framework, with any of the staticness or smoothness measures presented. Suppose $\mathcal{S}$ contains two SISO sub-functions, $w_1 = \mathcal{S}_1(z_1)$ and $w_2 = \mathcal{S}_2(z_2)$. A dispersion function which enforces that both are static, and equal, is, simply,

$$\mathcal{J}(\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}), \tag{3.139}$$

that is, just collect the two input-output records into one, and proceed as before. This simply tests whether the *union* of (z_1, w_1) with (z_2, w_2) is consistent as an input-output data record of a static operator.

3.4 Stochastic Criteria

As discussed in Chapter 3, the process of estimating z and w involves the simultaneous estimation of the disturbance signal e which accounts for undermodeling and measurement error in the models we consider. It is important to add appropriate constraints on the disturbance estimates that reflect the characteristics of these unmodeled influences. If not, any optimization procedure will use this degree of freedom to explain the part of the output measurement equation (3.3a) that depends on w, and to the extent that this occurs, the estimates of z and w will be thrown off. This will happen to some extent even when we do endeavor to constrain the problem, but we can try to minimize this effect.

What are relevant properties of e to consider? Probably the most important is its size, as measured in some norm. The size of the estimated noise should not be larger than the size of a typical realization of the noise signal during any given experiment. More generally we assume that e is generated by a random process (as discussed in Section 2.1) and require that estimates are consistent with the statistical properties of such a process. In the language of statistics, we can pose various hypothesis tests that we require any estimates to pass. If the estimate does not pass a given hypothesis test for the assumed distribution, then we decide the estimate is not consistent with having been generated by such a random process, and throw it out. Some tests to consider are those associated with the estimate's mean, variance, whiteness, and (un)correlation with other signals in the problem.

These criteria and tests can be used to constrain the search for estimates of unknown signals. As always, some of these tests are more or less amenable to different solution frameworks. Even if not included in the optimization problem, another way these tests can

be useful is to check the estimates that are computed. If they are not consistent with some of the statistical assumptions then perhaps constraints can be added to the problem, or the solution method can be modified, to arrive at better estimates.

The question arises, how does one discover the statistical properties of e? This is a hard question which we aren't prepared to answer systematically here, however can make a few comments. If it is possible, one experiment to perform is to observe the output of the device under test in some sort of steady-state experiment, for instance when the controllable inputs are held at zero or a constant value. In this case variability in the output may reflect the effect of unmodeled disturbances and measurement errors, and thus reveal something about their nature. Oftentimes it is reasonable to treat the noise process as though it satisfies certain assumptions. For instance that is is white, zero mean, or that it is uncorrelated with the input signal u. Often something is known about the nature of certain types of noise. For instance for position measurements that are generated with a potentiometer device it may be reasonable to assume the measurement error is normally distributed, whereas if the measurement is generated by an optical encoder then the error might be more accurately described by a uniform distribution. Finally, there is always trial and error; if one set of assumptions doesn't seem to produce good results, try another. For instance one can repeat the estimation process using various noise level constraints.

3.4.1 Mean

The sample mean of a vector v of length L is defined

$$\overline{v} := \frac{1}{L}\sum_{k=1}^{L} v(k). \tag{3.140}$$

If we believe the i'th disturbance signal e_i is a zero-mean process, then the sample mean of an estimate of it should be close to zero. This leads to constraints such as

$$-\delta_i \leq \overline{e}_i \leq \delta_i, \quad i = 1..n_e, \tag{3.141}$$

or with the parametrization of the consistent e as in (3.12b),

$$-\delta_i \leq \frac{1}{L}\sum_{k=1}^{L}(e_i^0(k) + K_{e,i}(k)f) \leq \delta_i. \tag{3.142}$$

Since the sample mean of a vector is a linear function of the components, these are $2n_e$ linear inequality constraints among the elements of e or the decision variables f. Another

possibility is to simply require the sample mean to equal zero. Linear equality and inequality constraints are typically easy for solvers to handle.

To make "close to zero" more precise and provide some guidance in choosing the constraint levels we can turn to standard statistical and decision theory considerations. For V a scalar IID stochastic process with zero mean and variance σ^2, its length-L sample mean $\overline{V}$ (a random variable) has expectation equal to zero and variance

$$\mathrm{Var}(\overline{V}) = \frac{\sigma^2}{L}. \tag{3.143}$$

Thus for a given interval about zero the sample mean falls inside it with greater and greater probability as L increases. In the Gaussian case where $V(t) \sim \mathcal{N}(0, \sigma^2)$, $\forall t$, the sample mean is itself Gaussian:

$$\overline{V} \sim \mathcal{N}(0, \frac{\sigma^2}{L}). \tag{3.144}$$

When the $V(t)$ are not necessarily Gaussian, as long as they are independent the central limit theorem tells us the distribution of the sample mean approaches this same Gaussian distribution as the sample length grows. In our problems L is typically large and we can assume the sample mean is well approximated by (3.144). Knowing the distribution of the sample mean, for a given confidence level $0 \leq \gamma \leq 1$ one can compute a confidence interval such that 100γ percent of the length-L realizations of V have a sample mean within this interval. A common choice is

$$I = [-\delta, \delta], \quad \delta = \frac{\sigma}{\sqrt{L}} z_{0.5+\gamma/2}, \tag{3.145}$$

where z_α is defined to be the α'th percentile of the unit normal distribution. That is to say, if Z is unit normally distributed, $Z \sim \mathcal{N}(0, 1)$, then $P(Z < z_\sigma) = \alpha$. Note that $P(|Z| < z_{0.5+\gamma/2}) = \gamma$.

Applying this to the problem at hand, suppose there is reason to believe the i'th disturbance e_i is IID, zero mean, with a given variance. Then given γ a δ_i can be computed such that the sample mean of realizations of such a process would fall within the interval (3.141) 100γ percent of the time. Given a length-L vector $\hat{e}_i$, this provides a way to evaluate the hypothesis "is the sample mean of $\hat{e}_i$ consistent with that of the assumed distribution of the i'th disturbance?" Namely if (3.141) is satisfied, then we accept the hypothesis.

Another possibility, which is useful when it's not possible or easy to derive the distribution of the sample mean based on the distribution of the underlying process, is to use the Chebyshev inequality.

Fact 3.13 (Chebyshev inequality) *Let X be a random variable with mean m_X and variance σ^2. Then for any $\delta > 0$,*

$$P(|X - m_X| \geq \delta) \leq \frac{\sigma^2}{\delta^2}.$$

Letting X be the sample mean of the zero-mean process V and using (3.143), we have

$$P(\overline{V} \notin [-\delta, \delta]) < \frac{\sigma^2}{L\delta^2}. \tag{3.146}$$

This provides another way to choose the interval δ in order to *guarantee* a given confidence level γ; set $\sigma^2/(L\delta^2) = \gamma$.

These results can be used to replace the choosing of δ_i for the sample mean constraint with the choosing of a confidence level, a more natural and meaningful number. It is reasonable to require that e_i meets this, for large γ. For example if realizations of the assumed distribution associated with e_i satisfy (3.141) 99% of the time, then it seems reasonable to require estimates e_i to also fall within the interval.

Any of these results can be easily adapted to a process with nonzero mean.

3.4.2 Variance

The variance of the noise signal, which is essentially the size of the noise estimates as measured by the l_2 norm, is perhaps the most important stochastic criteria to constrain.

One possibility is to follow the same approach of imposing constraints which enforce that the estimates are consistent with a distribution of a given variance.

A natural estimator of the variance of a stochastic process given a realization of the process is the *sample variance*. The sample variance of a vector v of length L is defined as

$$S^2(v) := \frac{1}{L}\sum_{k=1}^{L}(v(k) - \overline{v})^2, \tag{3.147}$$

where $\overline{v}$ is the sample mean as defined above.

Now if V is a scalar IID stochastic process with variance σ^2, the expectation of its length-L sample variance $S^2(V)$ is (nearly) equal to σ^2:

$$\frac{L}{L-1}E[S^2(V)] = E\left[\frac{1}{L-1}\sum_{k=1}^{L}(v(k) - \overline{v})^2\right] = \sigma^2. \tag{3.148}$$

(Some authors define the sample variance to be slightly different, as suggested by the middle expression, which is an unbiased estimator of variance. At any rate the two are essentially equal for large L.)

The variance of the sample variance is

$$\mathrm{Var}(S^2(V)) = \frac{(L-1)[(L-1)\mu_4 - (L-3)\mu_2^2]}{L^3}, \tag{3.149}$$

where μ_n is the n'th central moment of $V(t)$, and which simplifies to

$$\mathrm{Var}(S^2(V)) = \frac{2\sigma^4(L-1)}{L^2} \tag{3.150}$$

when V is Gaussian, $V(t) \sim \mathcal{N}(0, \sigma^2)$.

As we argued for the mean in the previous section, since the expectation of the sample variance equals the variance of the process, and its variance decreases with increasing sample length, it makes sense to confine the sample variance of the estimate of e_i to smaller and smaller intervals about the assumed variance of e_i as the data record length L increases. Analogously to (3.141), this results in optimization constraints of the form

$$S^2(e_1) \in [\sigma_i^2 - \delta_i, \sigma_i^2 + \delta_i], \tag{3.151}$$

where the choice of δ_i results in a certain confidence level.

For help choosing meaningful confidence intervals the following points may be useful. If V is IID and normal, then the scaled sample variance has a *chi-squared* distribution with $L-1$ degrees of freedom:

$$\frac{L}{\sigma^2} S^2(V) \sim \chi^2_{L-1}. \tag{3.152}$$

If V is zero-mean then the sample variance becomes

$$S^2(v) = \frac{1}{L} \sum_{k=1}^{L} v^2(k), \tag{3.153}$$

which is the sample mean of V^2, and if the process is IID (but not necessarily normal) then applying the central limit theorem one can derive the limit distribution of S^2 as $L \to \infty$ in terms of the mean and variance of V^2. Another alternative is to again use the Chebyshev inequality, taking advantage of the expression for the variance of the sample variance as in (3.149) or (3.150).

So we come up with bounds on the sample variance of the disturbance estimates e_i. Looking at the definition (3.147), the sample variance of e_i is essentially the 2-norm of e_i. Indeed if we assume the sample mean is small, as is often the case, we have

$$S^2(e_i) \cong \frac{1}{L} \|e_i\|_2^2 . \tag{3.154}$$

We often use $\|e\|_2^2$ in place of $S^2(e)$ to formulate the problems.

In either case, whether we use the sample variance $S^2(e_i)$ or 2-norm $\|e_i\|_2$, these considerations result in inequality constraints on convex quadratic functions of e_i. Or if we substitute the feasible-solution parametrization (3.12b), convex quadratic functions of f. These are readily handled in various solver software.

Practically speaking, when we include these constraints in an optimization problem and solve, the lower-bound constraints on the size of the disturbance estimates are typically not active. In problems where the size of e is constrained, the solver usually uses all of the e it can in order to minimize the objective, and the size of the estimated disturbance achieves the upper constraint bound. We can usually drop the lower bound, and (3.151) becomes

$$S^2(e_i) \leq \sigma_i^2 + \delta_i. \tag{3.155}$$

Having motivated these sample variance or two-norm bounds and the appropriate constraint levels to use with statistical considerations, let us now comment that at times it is easier in practice to essentially ignore these guidelines, and simply choose a bound, and use a constraint such as

$$\|e_i\|_2^2 \leq \alpha_i. \tag{3.156}$$

One possibility is to try the problem with several different choices of bound, and then decide which seem to produce the best results. This allows one to avoid getting into some of the details of the assumed distribution of the disturbance process and its variance.

The variance of a signal is one measure of its "spread". Other types of constraints that are related to the sample spread of the estimated disturbance are also possible. For instance if the disturbance signal e is assumed to be normally distributed on an interval I, then we might add the constraints

$$e(t) \in I, \quad t = 1..L. \tag{3.157}$$

This could be in place of, or in addition to, the variance constraints.

Let's pause for a moment here and look at some bigger-picture ideas. Many if not most traditional system identification methods fit models by essentially optimizing the error between the output as predicted in some way by the candidate model, and the measured output, to be as close to zero as possible. For instance, problems that boil down to linear regression do this, as do most parameter estimation approaches. Now if the measured output includes some noise then these methods will attempt to fit that noise, to the extent possible within the model structure chosen, so that for the particular data used the output is explained in terms of the input, along with as little noise as possible. This is not really what we intend to happen for then the model only "works" for that particular disturbance realization. What saves us of course is that usually the model structure is parametrized with few enough parameters that only models with smooth behavior are allowed, and the noise is averaged out instead of fitted. However if there are too many parameters then you *do* fit then noise. This is known as the bias/variance tradeoff. It is up to the designer to choose the right model parameterization, and number of parameters, so that this tradeoff is optimized in the final model. It is sometimes not clear though how many parameters are needed apriori, and a trial and error process is often involved to get it right.

If something is known about the level of noise involved in the problem then it may make more sense, and make the problem easier from an engineering standpoint, to simply add constraints to the problem so that the mismatch between the model's predictive power and the measured output does not *exceed* known limits on the size of the noise. But don't try to optimize this mismatch away to zero. Use the extra degrees of freedom to optimize other desirable model aspects (such as smoothness, consistency with prior knowledge of the system, etc). This is arguably a more direct approach to the bias/variance tradeoff, than to depend on the model parametrization to provide the correct level of smoothness.

3.4.3 Correlations

A common assumption is that the disturbance signals e_i are generated by a process which is independent of the past. Thus $e_i(t)$ is independent of $y(s)$, $u(s)$, $w(s)$, and $z(s)$ for $s < t$. Also $e_i(t)$ is independent of $e_j(s)$ unless $i = j$ and $s = t$. We may want to check if estimates are consistent with this assumption. With the additional assumption that e_i is stationary, this would mean that e_i is a white noise process. For instance if e_i is IID then it is necessarily white.

Correlation between random variables is often used as a check on their independence. A necessary condition that two random variables X and Y are independent is that they are uncorrelated, namely

$$E[(X - \mu_x)(Y - \mu_y)] = 0. \tag{3.158}$$

In fact, correlation measures the degree of *linear* dependence. It can be argued that higher-order correlation and other statistics are useful in detecting nonlinear relationships between two signals [4] [10]; we will not consider those here. In the case of normal random variables, uncorrelated is equivalent to independent.

Given samples of two random variables, it is straightforward to form estimates and tests of their correlation or lack thereof. Therefore a useful check or constraint to apply to estimated disturbance signals involves their correlation with other signals in the problem. The assumption that e_i is white also involves checking correlations.

From any probability or statistics book: if X and Y are two scalar discrete-time stationary stochastic processes, their *covariance function* is defined

$$\gamma_{xy}(h) := E[(X(t+h) - \mu_x)(Y(t) - \mu_y)], \tag{3.159}$$

where μ_x is the expected value of $X(t)$. Due to stationarity $E[X(t)]$ is independent of t, as is $\gamma_{xy}(h)$. The argument h is known as the *lag*. When X and Y are the same process this function is called the autocovariance, $\gamma_x(\cdot)$. A normalized version of the covariance is the *correlation function*, defined

$$\rho_{xy}(h) := \frac{\gamma_{xy}(h)}{\sqrt{\gamma_x(0)\gamma_y(0)}}. \tag{3.160}$$

Fun facts:

- The variance of X is found in the 0'th lag of its autocovariance function:

$$\mathrm{Var}(X) = \gamma_x(0). \tag{3.161}$$

- The magnitude of the correlation is bounded by 1 for all lags:

$$-1 \leq \rho_{xy}(h) \leq 1 \quad \forall h. \tag{3.162}$$

- A stationary random process X is said to be *white* if $\gamma_x(h) = 0$ for $h \neq 0$.

For two length-L vectors $x = (x(1), \ldots, x(L))$ and $y = (y(1), \ldots, y(L))$, their *sample covariance* is defined for $h = 0, 1, \ldots, L-1$ as

$$\hat{\gamma}_{xy}(h) := \frac{1}{L} \sum_{t=1}^{L-h} (x(t+h) - \overline{x})(y(t) - \overline{y}), \tag{3.163}$$

and for negative lags $\hat{\gamma}_{xy}(h) := \hat{\gamma}_{yx}(-h)$. As before $\overline{x}$ is the sample mean of x. The sample autocovariance of x is the sample covariance of x with itself,

$$\hat{\gamma}_{x}(h) := \frac{1}{L} \sum_{t=1}^{L-h} (x(t+h) - \overline{x})(x(t) - \overline{x}). \tag{3.164}$$

Note $\hat{\gamma}_x(0)$ is the sample variance of x. The *sample correlation* is

$$\hat{\rho}_{xy}(h) := \frac{\hat{\gamma}_{xy}(h)}{\sqrt{\hat{\gamma}_x(0)\hat{\gamma}_y(0)}}. \tag{3.165}$$

The sample autocorrelation reduces to

$$\hat{\rho}_{x}(h) := \frac{\hat{\gamma}_{x}(h)}{\hat{\gamma}_x(0)}. \tag{3.166}$$

The commonly used tests for correlation and whiteness are based on the large-sample asymptotic distribution of the sample covariance and correlation functions under certain assumptions [37] [25] [31]. Suppose X is a scalar white noise process. Then for the sample covariance and correlation as defined above, in the limit as L grows large,

$$\begin{bmatrix} \hat{\rho}_x(1) \\ \hat{\rho}_x(2) \\ \vdots \end{bmatrix} = \frac{1}{\hat{\gamma}_x(0)} \begin{bmatrix} \hat{\gamma}_x(1) \\ \hat{\gamma}_x(2) \\ \vdots \end{bmatrix} \xrightarrow{\text{d}} \mathcal{N}(0, \frac{1}{L} I), \tag{3.167}$$

i.e. a vector composed of the nonzero lags of the sample covariance converges in distribution to a zero-mean normal random vector with variance inversely proportional to L, i.e. in the limit the lags of the sample covariance are IID zero mean normal, with decreasing variance as L grows. Similarly if Y is another scalar stationary stochastic process independent of X, then

$$\begin{bmatrix} \vdots \\ \hat{\rho}_{xy}(-2) \\ \hat{\rho}_{xy}(-1) \\ \hat{\rho}_{xy}(0) \\ \hat{\rho}_{xy}(1) \\ \hat{\rho}_{xy}(2) \\ \vdots \end{bmatrix} = \frac{1}{\sqrt{\hat{\gamma}_x(0)\hat{\gamma}_y(0)}} \begin{bmatrix} \vdots \\ \hat{\gamma}_{xy}(-2) \\ \hat{\gamma}_{xy}(-1) \\ \hat{\gamma}_{xy}(0) \\ \hat{\gamma}_{xy}(1) \\ \hat{\gamma}_{xy}(2) \\ \vdots \end{bmatrix} \xrightarrow{\text{d}} \mathcal{N}(0, \frac{1}{L} I) \tag{3.168}$$

See [37] for proof of these asymptotic distributions, where it is required that Y is a "linear process".

With these asymptotic sample distributions in hand several tests for whiteness and independence of e_i with other signals can be applied. In the following let e represent a disturbance signal, and v any "other signal" whose correlation with e is being investigated. Some possibilities are

1. Test individual lags. Since (in the limit)

$$\frac{\hat{\gamma}_e(h)}{\hat{\gamma}_e(0)} \sim \mathcal{N}(0, L^{-1}), \quad h \neq 0, \tag{3.169}$$

a hypothesis test that the lag-h autocorrelation statistic of e ($h \neq 0$) is consistent with white noise, with confidence level γ, is to reject e unless

$$\frac{|\hat{\gamma}_e(h)|}{\hat{\gamma}_e(0)} \leq \frac{1}{\sqrt{L}}\, z_{0.5+\gamma/2}. \tag{3.170}$$

This inequality can be equivalently expressed in the form of two constraints that are quadratic in e,

$$\hat{\gamma}_e(h) \leq \frac{1}{\sqrt{L}}\hat{\gamma}_e(0)\, z_{0.5+\gamma/2} \quad \text{and} \quad -\hat{\gamma}_e(h) \leq \frac{1}{\sqrt{L}}\hat{\gamma}_e(0)\, z_{0.5+\gamma/2} \tag{3.171}$$

Similarly, a hypothesis test that the lag-h correlation statistic of e and v is consistent with that lag being uncorrelated is

$$\frac{|\hat{\gamma}_{ev}(h)|}{\sqrt{\hat{\gamma}_e(0)\hat{\gamma}_v(0)}} \leq \frac{1}{\sqrt{L}}\, z_{0.5+\gamma/2} \tag{3.172}$$

or equivalently

$$\hat{\gamma}_{ev}(h) \leq \frac{1}{\sqrt{L}}\sqrt{\hat{\gamma}_e(0)\hat{\gamma}_v(0)}\, z_{0.5+\gamma/2} \quad \text{and} \quad -\hat{\gamma}_{ev}(h) \leq \frac{1}{\sqrt{L}}\sqrt{\hat{\gamma}_e(0)\hat{\gamma}_v(0)}\, z_{0.5+\gamma/2} \tag{3.173}$$

If more than one lag is to be tested, the confidence level should be adjusted. Recall the probability that M independent events, each of probability β, all occur, is β^M. Thus if a total of M lags are to be tested, for an overall confidence level γ, the individual lags should be tested at level $\beta = \gamma^{1/M}$.

2. Test sums of lags.

Since the autocorrelation estimates (3.166) are asymptotically IID normal with mean zero and variance L^{-1} for $h \neq 0$, the sample mean of this quantity over M lags is

asymptotically normal with zero mean and variance $(LM)^{-1}$. This motivates the level-γ test for whiteness

$$\frac{1}{M}\left|\sum_{k=1}^{M}\hat{\gamma}_e(h_k)\right| \leq \frac{1}{\sqrt{LM}}\hat{\gamma}_e(0)\, z_{0.5+\gamma/2}, \tag{3.174}$$

or equivalently

$$\frac{1}{M}\sum_{k=1}^{M}\hat{\gamma}_e(h_k) \leq \frac{1}{\sqrt{LM}}\hat{\gamma}_e(0)\, z_{0.5+\gamma/2}, \quad -\frac{1}{M}\sum_{k=1}^{M}\hat{\gamma}_e(h_k) \leq \frac{1}{\sqrt{LM}}\hat{\gamma}_e(0)\, z_{0.5+\gamma/2}. \tag{3.175}$$

Similarly for cross-correlation estimates,

$$\frac{1}{M}\left|\sum_{k=1}^{M}\hat{\gamma}_{ev}(h_k)\right| \leq \frac{1}{\sqrt{LM}}\sqrt{\hat{\gamma}_e(0)\hat{\gamma}_v(0)}\, z_{0.5+\gamma/2}, \tag{3.176}$$

or

$$\frac{1}{M}\sum_{k=1}^{M}\hat{\gamma}_{ev}(h_k) \leq \frac{1}{\sqrt{LM}}\sqrt{\hat{\gamma}_e(0)\hat{\gamma}_v(0)}\, z_{0.5+\gamma/2}, \quad -\frac{1}{M}\sum_{k=1}^{M}\hat{\gamma}_{ev}(h_k) \leq \frac{1}{\sqrt{LM}}\sqrt{\hat{\gamma}_e(0)\hat{\gamma}_v(0)}\, z_{0.5+\gamma/2}. \tag{3.177}$$

Note that these formulations includes tests of single lags as a special case, when the sum is over just one lag.

3. Test sums of squares of lags.

   An approach advocated in [25] is to use the sum of squares of a number of lags of the autocorrelation to form a test for whiteness. In the limit, this is the sum of the squares of independent normal random variables, which results in a chi-squared random variable,

$$\frac{\sqrt{L}}{\hat{\gamma}_e(0)}\sum_{k=1}^{M}\hat{\gamma}_e(h_k) \xrightarrow{\mathrm{d}} \chi^2_M, \tag{3.178}$$

   and this can be used as the basis of a hypothesis test, this time based on the chi-squared distribution.

For a disturbance estimate $\hat{e}_i$, (3.170), (3.174), and (3.178) can be used to test its whiteness. Any or all may be included in the estimation procedure. The summed versions (3.174) and (3.178) have the advantage of taking several lags into account with a single constraint with essentially no added complexity, while testing the individual lags as in (3.170) gives finer control. The expressions (3.172) and (3.176) can be used to test

correlation of e_i with other signals in the problem. For instance if we can assume the disturbance is independent of the past, as is often the case, then the expectation of $\hat{\gamma}_{e_i v}(h)$ should be zero for $h > 0$ and v any component of y, u, z, or w. Sometimes the assumption holds for $h < 0$ as well, for example $e_i(t)$ is independent of $u_j(s)$ for all s, t, in which case we test that $\hat{\gamma}_{e_i u_j}(h)$ is small for $h < 0$ as well. We need to be careful here, where feedback loops in the system can mean that certain signals are dependent on past values of the disturbance.

What about properties of these types of constraints, and the tractability of including them in convex programming problems? Looking at (3.164), the sample autocovariance is nonconvex quadratic in the elements of x, while the sample covariance (3.163) is bilinear in the elements of x and y. Thus tests like (3.171) are nonconvex quadratic in e, or nonconvex quadratic in f when the feasible parametrization of Section 3.2.1 is used. The cross-correlation test (3.173), thinking of both e and v as unknown quantities to be estimated parametrized as in Section 3.2.1, are nearly quadratic if not for the square root term on the right hand sides. In practice if we impose variance constraints on e such as (3.155) or (3.156), and can assume the estimate achieves the upper bound, then we can replace $\hat{\gamma}_e(0)$ in these expressions with that value. A greater simplification happens when testing the correlation of e_i with a signal v that is known. In this case the sample correlation $\hat{\gamma}_{e_i v}(h)$ is a linear function of f and $\hat{\gamma}_v(0)$ is a fixed deterministic quantity, and the tests (3.171) and (3.175) become pairs of linear inequality constraints in f. Of course linear inequality constraints are easily handled in convex programming problems. With the nonconvex quadratic constraints that appear otherwise we can either attempt to solve the nonconvex programming problem directly, look for convex approximations and relaxations, or be content to use them for *a posteriori* checks on estimate quality.

Chapter 4

Formulations

In previous chapters we laid out a problem involving estimation of unknown signals in an interconnected model structure, one involving unknown static functions. This particular problem was motivated by a system identification task, where unknown static operators such as these are to be estimated. A general approach to producing estimates was presented in Chapter 3, which comes down to formulating an optimization program and estimating certain unknown signals in the problem based on qualities good estimates should posses. Sections 3.3 and 3.4 introduced a number of quantities that one may want to use in order to achieve this. Recall that these measure and constrain various staticness, smoothness, and stochastic qualities of estimates. The goal has been to find measures which accurately capture qualities of interest, and at the same time are simple enough to be usable in a computational setting. In this chapter we look at specific formulations composed of these elements, and issues involved in optimizing them. Examples are used to illustrate these choices and to show typical results.

4.1 Assembling Programs

4.1.1 Review Of Elements

As a review, and for the sake of having a reference with consistent notation, many of the optimization elements from the previous chapter that we will be using here are listed in Table 4.1. Most of the elements can be used either as objectives, or as constraints by picking an appropriate constraint level; an appropriate expression for each is shown.

elmnt	objective form	constraint form
$\mathcal{J}$ ($\mathcal{J}_w$)	$\min \sum_{k=1}^{n_w} (\Gamma_{w,k}^2 \lVert D^1 P^{z_k} w_k \rVert_2^2 + \Gamma_{z,k}^2 \lVert D^1 P^{z_k} z_k \rVert_2^2)$	$\Gamma_{w,k}^2 \lVert D^1 P^{z_k} w_k \rVert_2^2 + \Gamma_{z,k}^2 \lVert D^1 P^{z_k} z_k \rVert_2^2 \leq 1, k = 1..n_w$
$\mathcal{J}_2$ ($\mathcal{J}_{2w}$)	$\min \sum_{k=1}^{n_w} (\Gamma_{w,k}^2 \lVert D^2 P^{z_k} w_k \rVert_2^2 + \Gamma_{z,k}^2 \lVert D^2 P^{z_k} z_k \rVert_2^2)$	$\Gamma_{w,k}^2 \lVert D^2 P^{z_k} w_k \rVert_2^2 + \Gamma_{z,k}^2 \lVert D^2 P^{z_k} z_k \rVert_2^2 \leq 1, k = 1..n_w$
$\mathcal{J}_{\mathrm{A}}$ ($\mathcal{J}_{\mathrm{TV}}$)	$\min \sum_{k=1}^{n_w} \sum_{t=1}^{L-1} \left((\Gamma_{z,k} D_t^1 P^{z_k} z_k)^2 + (\Gamma_{w,k} D_t^1 P^{z_k} w_k)^2 \right)^{1/2}$	$\sum_{t=1}^{L-1} \left((\Gamma_{z,k} D_t^1 P^{z_k} z_k)^2 + (\Gamma_{w,k} D_t^1 P^{z_k} w_k)^2 \right)^{1/2} \leq 1, k = 1..n_w$
$\lVert w \rVert_2$	$\min \sum_{k=1}^{n_w} \Gamma_{w,k} \lVert w_k \rVert_2^2$	$\lVert w_k \rVert_2^2 \leq \alpha_{w,k}^2, k = 1..n_w$
LipCP		$\lvert P_{t+1}^z w_k - P_t^z w_k \rvert \leq \lambda_k \lvert P_{t+1}^z z_k - P_t^z z_k \rvert,\ t = 1..L, k = 1..n_w$
LipAP		$\lvert P_t^z w_k - P_s^z w_k \rvert \leq \lambda_k \lvert P_t^z z_k - P_s^z z_k \rvert,\ s, t = 1..L, k = 1..n_w$
$\lVert e \rVert_2$	$\min \sum_{k=1}^{n_e} \Gamma_{e,k} \lVert e_k \rVert_2^2$	$\lVert e_k \rVert_2^2 \leq (\alpha_{e,k}^2 := \Gamma_{e,k} L \sigma_{e_k}^2), k = 1..n_e$
Corr		$\lvert \sum_{h \in H} \hat{\gamma}_{e_k v}(h) \rvert \leq \sqrt{\frac{\lvert H \rvert}{L}} \sqrt{\hat{\gamma}_e(0) \hat{\gamma}_v(0)} z_{.5+\gamma/2}$, for some k, H

Table 4.1: A collection of elements for use in formulating estimation problems.

We stick to staticness and smoothness measures for SISO operators. We generalize this to $\mathcal{S}$ that may be collections of several SISO $\mathcal{S}_k,\ k = 1..n_w$. To form staticness objectives out of the SISO measures we take weighted combinations of the measures for individual $\mathcal{S}_k$, and to use them as constraints we pick individual constraint levels for each of the $\mathcal{S}_k$; this was previously mentioned in Section 3.3.5. In Table 4.1 the weights, or individual constraints, are specified by the design variables Γ_w and Γ_z; these also allow for a different relative weighting of the w- and z-parts of the expressions. Note that the $\mathcal{J}_w$, $\mathcal{J}_{2w}$, and $\mathcal{J}_{\mathrm{TV}}$ measures can be obtained from $\mathcal{J}$, $\mathcal{J}_2$, and $\mathcal{J}_{\mathrm{A}}$ respectively, by setting $\Gamma_z = 0$. Similarly, the Lipschitz constraint level for $\mathcal{S}_k$ is specified by choice of λ_k.

The size of the constraint on the noise level $\|e_k\|_2$ is represented in the table by $\alpha_{e,k}$. Also perhaps more naturally we can work in terms of a weighting $\Gamma_{e,k}$, related to $\alpha_{e,k}$ as $\alpha_{e,k}^2 = L\,\Gamma_{e,k}^2\sigma_{e,k}^2$, where $\sigma_{e,k}^2$ is the variance of the noise process and L is the length of the (scalar) signal e_k. For instance by specifying $\Gamma_{e,k} = 1$, or equivalently $\alpha_{e,k}^2 = L\sigma_{e,k}^2$, and assuming zero-mean estimates, we are constraining the sample variance of e_k equal to the process variance (this will be a common choice). (Whether or not the process variance is known is another question.)

At the bottom of the table, for the constraints associated with correlations between signals, when we also use a noise level constraint like $\|e\|_2 \leq \alpha_{e,k}^2$, it is often a good approximation to replace $\hat{\gamma}_e(0)$ with $\alpha_{e,k}^2/L$. Then when v is a known signal this lets us use a constant for the constraint level in the optimization formulation, in place of an expression that depends on decision variables in a nonconvex nonquadratic way.

Some of the elements are more naturally used as either an objective or constraint. For instance since for a static operator the dispersion $\mathcal{J}$ decreases with increasing data density, it is hard to choose a meaningful constraint level independent of the data record, and thus it is often easier to minimize this property. On the other hand arclength and total variation are properties of the function independent of the data, and they don't decrease with increasing density, and so it makes sense to constrain $\mathcal{J}_{\mathrm{A}}$ or $\mathcal{J}_{\mathrm{TV}}$ if there is apriori knowledge about the total variation or arclength of $\mathcal{S}$. Lipschitz criteria are usually considered as constraints, however in principle one could minimize the maximum Lipschitz constant of the scatter plot (if z is fixed then this is a convex problem).

Regarding the noise level, $\|e\|_2$. In our experience it tends to be better, from a usability standpoint in terms of how easy it is to find formulations that produce good estimates, and perhaps makes more sense as well, to constrain this quantity, as opposed

to minimizing it. This may be counterintuitive to those familiar with more "traditional" methods in system identification such as maximum-likelihood parameter estimation, or basic least-squares fitting. A canonical description, in a stochastic setting, of what is happening in these methods is that given a parametrized model set which includes a noise model, parameter and noise estimates are sought which are consistent with experimental input-output data. It is then argued that the best parameter estimate is the one associated with the smallest noise estimate, in other words the noise estimate is indeed explicitly minimized, the reasoning being that for noise that is assumed to be e.g. a Gaussian process, then smaller estimates are more likely as realizations of the process.

An issue that is often raised here is known as the bias-variance problem (discussed previously in Section 1.4. Although a low-variance noise is more likely, the actual realization during a given experiment does not necessarily have small size. In fact, the higher the variance of the process, the more *unlikely* that the realization will have a sample variance smaller than a given number. What ends up happening in maximum likelihood parameter estimation is that any flexibility in the parametrization is used to minimize the size of the estimated noise signal. In a given experiment, to the extent that this level gets minimized below that of the actual experiment noise this is undesirable, since now the actual noise is being partly explained using the parameters of the model, and not the estimated noise (it is "fitting the noise"). The higher the number of parameters the more this overfitting is possible, resulting in a high-variance estimator. A more ideal formulation would be to *constrain* the size of the estimated noise, and then use criteria based on other knowledge about the problem to locate a best parameter estimate from the remaining feasible parameter choices. In our particular problem the number of parameters being estimated is quite large, being on the order of the length of the data record, and therefore the noise-fitting issue is an especially important consideration. Hence the comment that it may be more straightforward, in producing good estimates, to constrain $\|e\|_2$, rather than minimize.

Any of the formulations to be posed include the linear system constraints described in Section 3.2, and when solving we generally use the parametrization of feasible solutions to this constraint that are described in the following sections. Finally we note once again that for the large decision variables that the formulations use, we are limited as to what classes of program we can actually solve. In general we look for optimization programs that are convex, and linear or quadratic programs are ideal, including least squares problems (including Kalman filtering) and quadratically-constrained quadratic programming (QCQP).

Large semidefinite programming (SDP) or linear matrix inequality (LMI) problems, are also feasible. With these we can compute approximate solutions for large possibly-nonconvex quadratic programs (NQCQP), which can be useful for us. Other more general options are methods for minimizing smooth functions, and finally the very general nonlinear optimizers often involving randomized searching.

4.1.2 Fixed z vs. Free z

Whether or not the z signal depends on the decision vector f has important implications for many of the staticness elements regarding convexity and other characteristics that are important when doing optimization. In Section 3.3 we saw that many of the staticness operators are nonconvex functions of z. For $\mathcal{J}$, $\mathcal{J}_w$, $\mathcal{J}_2$, $\mathcal{J}_{2w}$, $\mathcal{J}_{\mathrm{A}}$, and $\mathcal{J}_{\mathrm{TV}}$ the difficulty is the sorting (or, in higher dimensions, the triangulation) matrix, P^z, which is nonconvex, nonquadratic, and discontinuous in z (and hence P^{z_f} is discontinuous in f). Lipschitz constraints are quadratic, but once again nonconvex, in z.

If z is measured or otherwise known with sufficient accuracy then we do not have to estimate it and can leave it out of the decision variables of the estimation optimization formulation. The dependence on f is then convex and benign for these staticness measures.

Otherwise we consider z as unknown. In this case the $\mathcal{J}_*$ and Lipschitz constraint functions are nonconvex and not handled by solvers that are efficient for the length of the decision vectors we want to consider. If we could only use those staticness measures in systems where z is fully known this would obviously impose quite a restriction on the generality of those formulations involving those elements. Therefore approximate solution methods or relaxations are needed for the problem we would ideally like to solve We can list three options in this situation:

1. Settle for a formulation which uses only criteria/constraints which are "nice" functions of f. This dramatically reduces the available options. One option still available here can be interpreted as a Kalman smoothing operation, or as an ill-posed inversion problem with a regularization criteria. This is minimization of the two-norm of w subject to a two-norm constraint on e, or minimization of a weighted combination of these.

2. Bootstrap. This is the label we use to refer to an iterative process in which each iteration produces new estimates of the unknowns e, w, and z, which are in turn used

to build the estimation formulation to produce the next iterate. To proceed, those parts of the criteria/constraints you would like to use that cause them not to be nice functions of f (usually this is the sort operation) are fixed at the current iteration's estimate of z, leaving an optimization problem which can be formulated as a convex cone problem. This is then solved, producing estimates for the next iteration. More of the staticness elements fit into such a framework, and various combinations can be used. We have found this family of formulations often works reasonably well, and is what we generally try first when z is fundamentally unknown.

A variation for the measures with sorting is to fix just P^z, and let the remaining dependence on z be free since it is convex.

3. Reformulate the problem as one that is not quite a convex cone problem, but can be *relaxed* into one. For instance this is possible with formulations that involve criteria/constraints that are quadratic, but possibly nonconvex. For these problems the so-called S-procedure (Boyd et al. 1994) may be employed to produce bounds on the optimal value, and quantities produced in solving this problem can be used in a guided search for suboptimal feasible solutions to the original problem (known as *primalization* (Goemans and Williamson 1995). Although this method produces larger problems and so is restricted to shorter data record lengths, and once again the set of available problem formulations is quite small since all criteria/constraints must be quadratic, However there are three main difficulties: 1) answer are conservative and not necessary as well as sufficient; 2) these formulations produce a bound on the optimal value, when what we really want to find in the current problem is estimates (perhaps suboptimal), and are not so interested in that actual objective level; and 3) the size of the optimization problem can grow dramatically. Initial trials have shown that this technique has some merit.

We note that a common situation in mechanical systems is a reaction force due to some sort of displacement. In some cases the reaction force as a function of the displacement is a nonlinear, unknown function to be identified. Typically it is more straightforward and practical to measure displacement than forces. If the displacement during an experiment is recorded, but not the reaction force, then we have the situation of an unknown function with measured input and non-measured output. For instance a common simple friction model is one that expresses frictional force as a function of sliding velocity; velocity is often

easier to sense than friction force. Another instance of a such a nonlinearity may be a gear with backlash; it's easy to monitor the angle of the two gears, but harder to monitor the reaction force.

4.1.3 Multiple Experiments

In identifying embedded static operators it's not hard to incorporate input/output data from multiple experiments. In fact it's possible to combine dissimilar model structures. This is useful when there are data records from multiple experiments and one has reason to use model structures for the experiments which differ but contain common static elements. Perhaps various configurations of a physical system were tested, each containing a physical part which is to be represented by a static function. Better estimates of common static elements may be obtained by combining the information contained in the multiple data records.

It may be especially important to be able to incorporate information from multiple experiments when dealing with nonlinear systems. For instance a nonlinear system can have multiple stable equilibria. Often a particular experiment will operate in the neighborhood of one of these equilibria, which means the signals stay in regions near their equilibrium values. If there are static maps to be identified this might result in some information about the map over some portion of its domain of interest, and little information over some other portion that corresponds to another equilibrium point. Other, separate, experiments might provide more information about these regions, and if we can combine experiments this will allow a more accurate identification of the map over the entire domain of interest.

The extension to multiple experiments is straightforward. Each experiment adds its own set of unknown signals and the approach once again is to form an estimate for each of the unknowns. Let v^i denote a signal associated with experiment i. So for instance in experiment 1 e^1 and w^1 may be unknown, and experiment 2 has unknowns e^2, w^2, and z^2. this case we are seeking to estimate the union of these, $(e^1, w^1, e^2, w^2, z^2)$.

As before the linear dynamic system associated with experiment i is a system of linear constraints among $x_1^i, u^i, e^i, w^i, y^i$, and z^i. Taken together they are a larger set of equations in the collected unknowns, and as such a parametrization of the consistent solutions in terms of a free parameter exists, just as was the case in Section 3.2 when we were considering a single experiment. One way to get such a parametrization is to "stack"

the parametrizations for the individual experiments as described in that section.

The criteria associated with the static operator(s) also extends easily. The situation is much the same as it was for repeated static elements in Section 3.3.5. In Section 3.3, each of the criteria we considered for evaluating the degree of consistency of an input-output pair (z, w) with the required properties of $\mathcal{S}$ is a function of the collection of input-output pairs $\{(z(t), w(t))\}_{t=1}^{L}$. The point is, the criteria do *not* depend on their time ordering. One can think of the criteria as really being functions of the scatter plots associated with the $z-w$ records. The fundamental reason for this is that we are talking about *static* operators, for which in a sense the time ordering of the input doesn't matter.

In light of this if by performing additional experiments more input-output points are obtained for a given static element, we should simply union those with any existing i-o pairs, and consider the consistency of the larger set with the required properties of $\mathcal{S}$. The procedure is, first make a list of the unique elements found in the collection of static operators $\mathcal{S}^i$. For example if $\mathcal{S}^1$ contains two block-diagonal elements, a two-input $w_1^1 = f(z_1^1, z_2^1)$ repeated once and a single-input $w_2^1 = g(z_3^1)$ repeated twice, and $\mathcal{S}^2$ consists of one block-diagonal element $w_1^2 = g(z_1^2)$ where $g(\cdot)$ is the same function as in experiment 1, then there are two unique elements in the unioned experiment, f and g.

Then, whether for the purpose of analyzing staticness, repeated elements, or smoothness properties, for each unique static element we now consider the union of its input and output from all the experiments where that element was present. Note that for instance the first input may be measured in some experiments and unknown and to be estimated in others. In any case the static operator criteria end up being functions of the free parameter which are then constrained or minimized in the process of computing estimates.

The criteria for the stochastic properties of the estimates of noise and disturbance signals also extend to multiple experiments. The noise signals from each of the experiments will have an assumed joint probability distribution with the other signals involved in the experiments, and these provide optimization criteria in the ways described in Section 3.4. In particular the stochastic signals from one experiment will usually be assumed statistically independent of signals in different experiments.

To summarize it's straightforward to incorporate information from multiple experiments to identify static model components. The main drawback to using more and more experiments is that since there are more unknowns and constraints the size of the optimization problem grows. This becomes the limiting factor.

4.2 Example: Bungee

Consider a mass m suspended by an elastic cord from a point whose vertical position u we can control (the upward direction corresponds to increasing value of position). A noisy measurement $y = x + e_1$ of the vertical position x of the mass is available. The noise is IID zero mean normal with variance $\sigma_{e_1}^2$. The model includes a gravity force $F_g = -mg$, viscous damping c on the motion of the mass, and a restoring force due to stretch in the cord. The restoring force $F_r = k(u-x)$ is a nonlinear function of the length $u-x$ of the cord; when the cord is stretched it acts as a hardening spring after an initial linear elasticity regime, and when the cord is slack it provides no restoring force. Finally there is an acceleration disturbance e_2 with variance $\sigma_{e_2}^2$. Discrete-time equations of motion come from discretizing the second-order continuous-time ODE using Euler's method ($x_1 := x, x_2 := \dot{x}$) with sample time $T = .05$ seconds. They are

$$x_1(t+T) = x_1(t) + Tx_2(t) \tag{4.1}$$

$$x_2(t+T) = x_2(t) + T\left[-cx_2(t) + w(t) + e_2(t)\right] \tag{4.2}$$

$$y(t) = x_1(t) + e_1(t) \tag{4.3}$$

$$w(t) = k(z(t)) - g =: k_e(z(t)) \tag{4.4}$$

$$z(t) = u(t) - x_1(t), \tag{4.5}$$

where z and w are the input and output of $k_e(\cdot)$, respectively, and the "effective" restoring force $k_e(\cdot)$ includes gravity. Note that z is nearly known since $u-y = z-e_1$ and presumably the level of the noise signal e_1 is low. However the restoring force w is not measured; the main problem is to estimate this quantity.

Following the steps laid out above, the first is to collect input-output data $(\bar{y}, \bar{u})$. For this example we simulate input-output data with the input u a lowpass-filtered uniformly distributed white noise, e a Gaussian white noise, and initial displacement $x = -8$. We generated data for three different noise levels, $\sigma_e \in \{0.05, 0.2, 1\}$, in order to see the effect of noise on the problem; there is a common unit-variance noise for the three cases, which is then scaled by σ_e. All other details of the simulation are held fixed throughout. The values of e, w, and z achieved during the simulation are similarly notated as $\bar{e}$, $\bar{w}$, and $\bar{z}$.

Figure 4.1 shows 400 samples of typical simulated signals, for the $\sigma_e = 0.2$ noise level. Note that the most negative the w signal ever gets is 9.8, when the cord is slack and force on the mass is due to gravity alone. The last panel is a scatter plot of w versus z; this

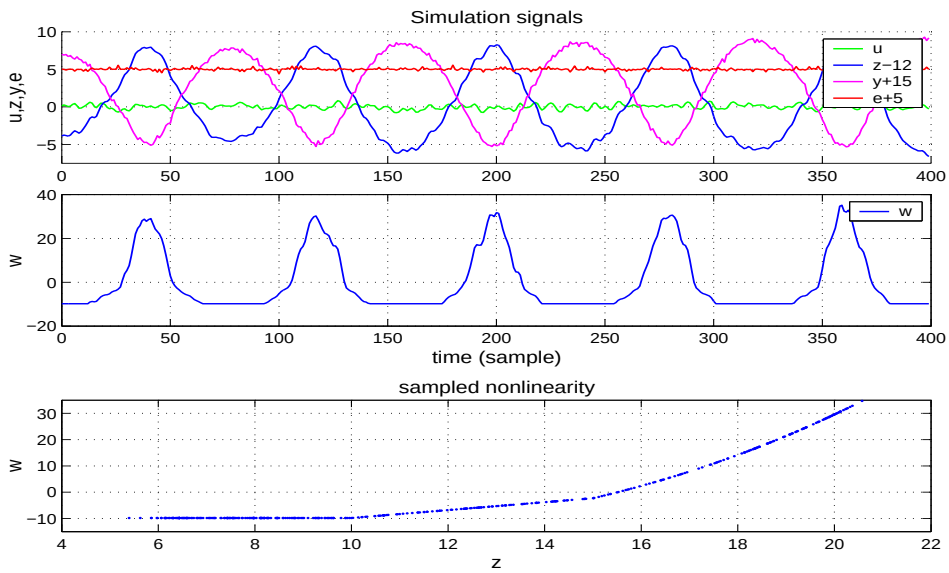


Figure 4.1: Typical simulation data for the bungee example.

is where the function $w = k_e(z)$ was sampled by this data record. This shows the shape of the nonlinearity; on it can be found the length when the cord is taut and not stretched (10), when it transitions from a linear to a cubic hardening spring (15), and the constant force due to gravity (-9.8).

Using this data along with knowledge of the linear system, the next step is to compute a parametrization of the consistent solutions (e^f, w^f, z^f), with f the free parameter. We follow the ideas in Sections 3.2.3 and 3.2.4 to find a parametrization such as (3.12).

Next we assemble various criteria such as those in Table 4.1 for estimating w and z, and solve. To get started we consider a formulation in which the objective is to minimize the dispersion $\mathcal{J}$ of the $\hat{z}$, $\hat{w}$ estimates, while constraining the norm $\|\hat{e}\|_2$ of the estimated noise to some value near its assumed variance. This formulation is among the simplest that makes use of one of the staticness criteria, and tends to gracefully accommodate a wide variety of situations.

As z, the cord length, is not fixed, i.e. z^f is indeed a function of the decision variable f, we look to one of the solution approaches described in Section 4.1.2. In this problem, with $\mathcal{J}$ involving the sort operator, the only choice described there which is applicable is

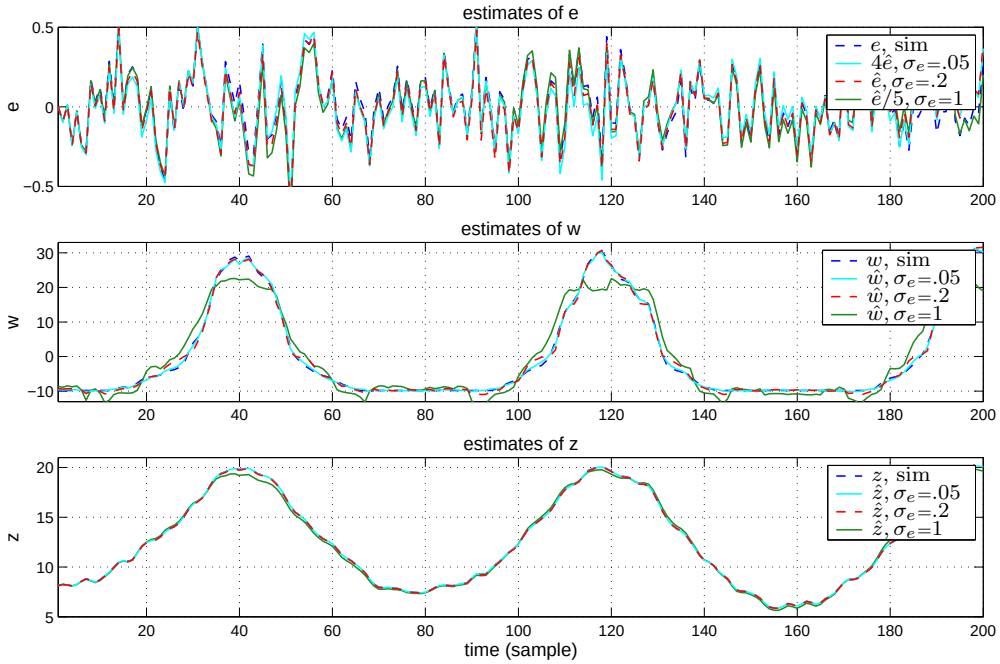


Figure 4.2: Estimated e, w, and z signals using 400 points, for three noise levels.

the bootstrapping method. Namely, at iteration k fix $P = P^{z^{f^{(k)}}}$, and solve

$$f^{(k+1)} = \begin{array}{c} \underset{f}{\text{argmin}} \quad \Gamma_w^2 \|D^1 P z^f\|_2 + \Gamma_z^2 \|D^1 P w^f\|_2 \\ \text{subject to} \quad \frac{1}{L}\|e^f\|_2^2 \leq \Gamma_e \, \sigma_e^2. \end{array} \tag{4.6}$$

This involves choosing an initial value for $f^{(0)}$. Here Γ_e is a design variable used to choose the noise constraint relative to the actual simulation noise variance σ_e^2. Other user choices are Γ_z, Γ_w, and what portion of the data set to use. We will use the first L samples of the data sets, where L will vary depending on the point being illustrated. To compute solutions we use SeDuMi [36], an efficient solver for convex self-dual cone problems, including (4.6) which is a quadratically-constrained quadratic program (QCQP).

Figure 4.2 shows three sets of estimates of the e, w, and z signals computed using the first 400 data points, each set corresponding to one of the three simulation noise levels. These use $\Gamma_e = \Gamma_w = \Gamma_z = 1$ for design weights. In particular, this means the size of the estimated noise is constrained to be equal to the variance of the process that generated the noise. These estimates are the eighth iterate of the bootstrap procedure. The initial (zero'th) iterate of the decision variable, $f^{(0)}$, was taken to be zero (and so the initial estimates $\hat{w}^{(0)}$, etc. are the particular solutions w^0, etc, as described in Section 3.2). In

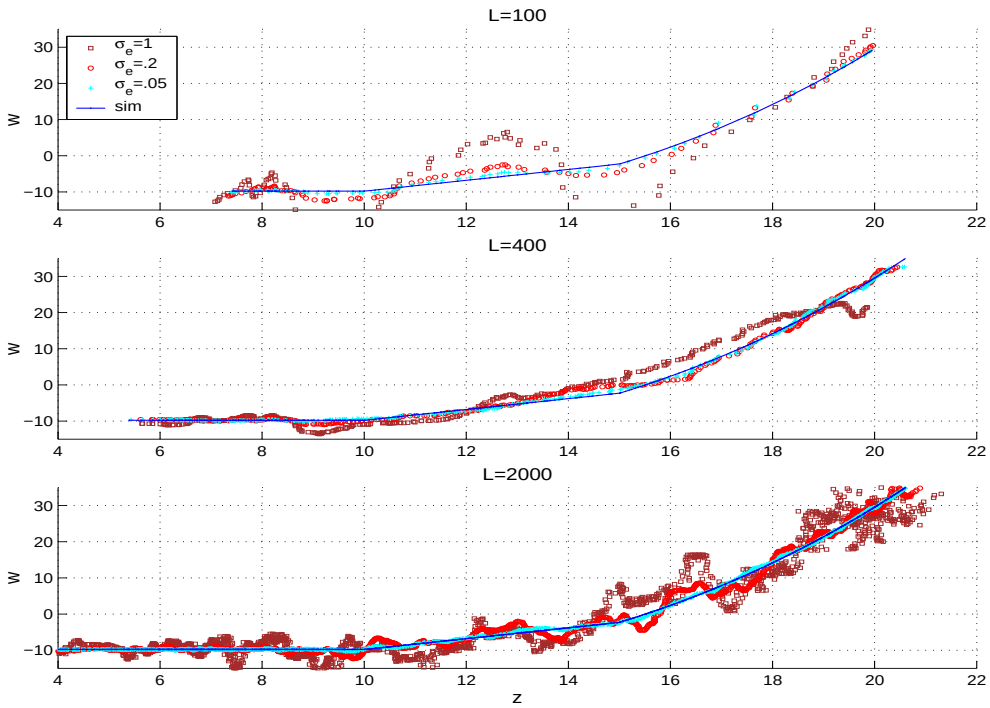


Figure 4.3: Scatter plots of estimated w and z, for three noise levels and different amounts of data.

this figure the estimates, including those of the noise signal, appear fairly accurate, except perhaps for the estimate of w using the data with the highest noise level. In Figure 4.3 are scatter plots of $\hat{w}$ vs $\hat{z}$, showing, for each of the three noise levels, estimates using 100 data points, 400 points, and 2000 points.

First of all, these indicate that the optimization problem we have formulated is doing a reasonable thing. From the estimates, and especially from the better ones, we get a fairly clear picture of the shape of the nonlinearity. Further, as we expect, the estimates get progressively better when there is less noise in the data. However the trend versus data record size is not as clear. In this example estimates improve greatly when using 400 points as opposed to 100, but then for the even longer record they seem to not be as good. Estimates for some other data lengths are shown in the top panel of Figure 4.4, this time for the $\sigma_e = 0.2$ data record only. Speaking roughly, estimates tend to improve with larger data lengths here until around $L = 400$ or $L = 800$, but then they become worse again.

Let's take a closer look at what's happening during the bootstrap iteration. Figure 4.5 shows the estimates after five of the iterates, using 800 points of the record from the $\sigma_e = 0.2$ data set. Namely they are the estimates corresponding to $f^{(k)}$ after 1, 2, 3, 5, and

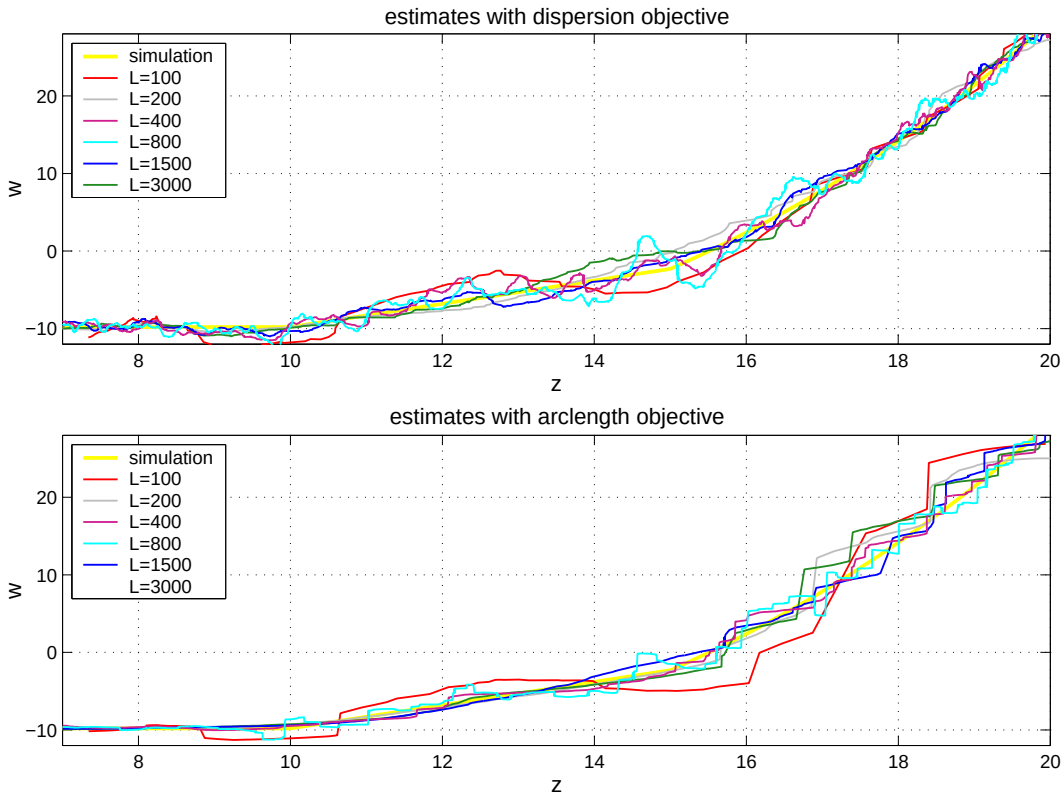


Figure 4.4:

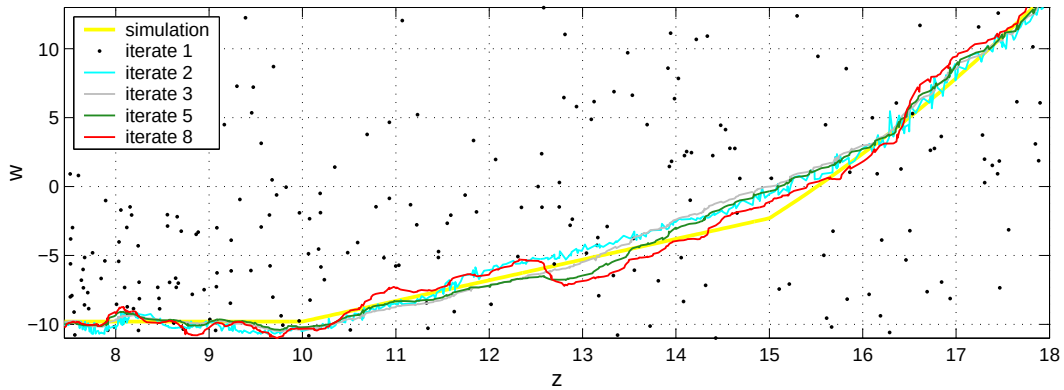


Figure 4.5: Five of the bootstrap iterates. $\sigma_e = 0.2$, $L = 800$

iterate	$\mathcal{J}$ **achieved**
0	37.45
1	632.31
2	13.87
3	5.08
4	4.14
5	4.70
6	4.84
7	5.14
8	4.72

Table 4.2: Dispersion objective for bootstrap iterates ($L = 800$, $\sigma_e = 0.2$)

8 iterates. We see the first iterate is very cloudy, wile the second is a vast improvement. This iterate produces a relatively "noisy" or "fuzzy" scatter plot, with lots of high-frequency jitter, as compared to the later estimates. The later iterates could be characterized as getting smoother, but progressively more biased, in the sense that the average level of w over windowed portions of the z axis are farther from the true function. The estimated scatter plot sort of "meanders". The dispersion objectives that each of the iterates achieves is shown in Table 4.2. Initial reductions of dispersion are large, followed by a lower level of bouncing around a smaller cost. Since we have the luxury of knowing the true signals, we can compute their dispersion cost: it is 3.54. We should note that the variance of the realization of the noise in this experiment is slightly larger than the noise constraint used in the estimation; in the estimation formulation we used $\alpha_e^2 = \Gamma_e L \sigma_e^2 = 32$, whereas in the simulation $\|\bar{e}\|_2$ is approximately 32.4044. An interesting thing to try is to start the bootstrap iteration at the simulation signals (set $e^0 = \bar{e}$, $w^0 = \bar{w}$, $z^0 = \bar{z}$, and $f^{(0)} = 0$), and with the correct noise constraint for $\bar{e}$. With this setup, the dispersion of the first 8 iterates is shown in Table 4.3. We see there is an initial *increase* in the dispersion! Later iterates (at least the first 50) move around between 4.1 and 5.2. Finally, for the same formulation, in Figure 4.6 we show four iterates (0,1,2,8), for each of 7 bootstrap trials starting from random initial f, along with the trial starting from the exact simulation data. We see that although the initial w^f and z^f are completely different, the bootstrap iteration converges to nearly the same estimate in each case (and nearly the same estimate as the previous formulation with the slightly smaller noise constraint produced). This seems to be a limit point for the bootstrap algorithm, one which is not sensitive to initial condition. For the original problem of minimizing dispersions subject to the noise constraint, which we are

iterate	$\mathcal{J}$ **achieved**
0	3.54
1	4.54
2	4.48
3	4.53
4	4.89
5	5.22
6	5.50
7	5.39
8	5.18

Table 4.3: Dispersion objective for bootstrap iterates ($L = 800$, $\alpha_e^2 = 32.4044$, simulation initial conditions)

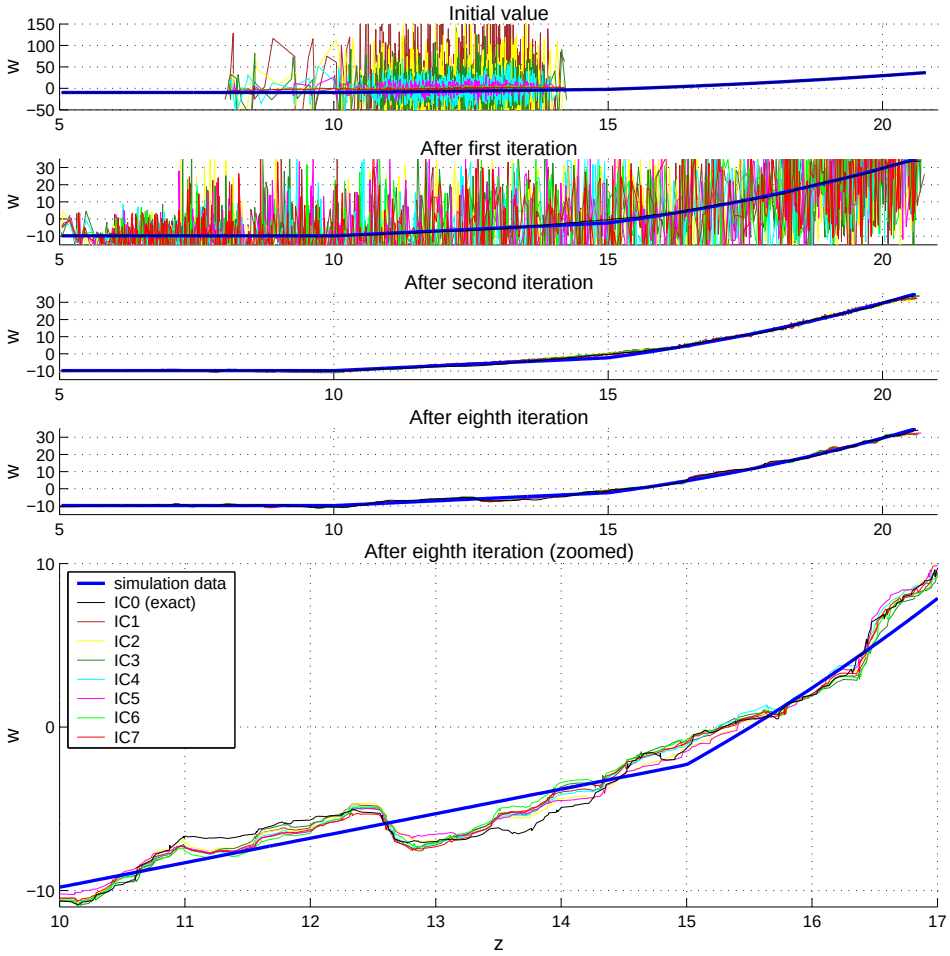


Figure 4.6: Four iterates from each of seven bootstrap sequences starting from random $f^{(0)}$, and one starting from exact simulation signals.

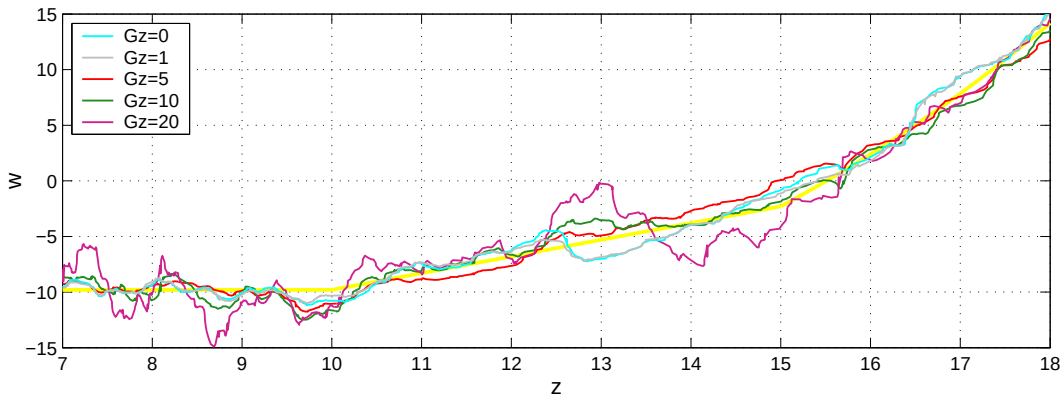


Figure 4.7: Scatter plots of estimates using various Γ_z.

employing the bootstrap iteration to solve, we have seen that there is at least one estimate that beats this limit. Namely the simulation is feasible, and has a smaller dispersion.

Hence the bootstrap iteration is not finding the optimal point. Our experience with this and other examples has been that the iteration is generally seen to robustly produce reasonable, but not optimal, estimates. The sequence of estimates produced by the iteration is typically not monotonic in in terms of a norm such as $\|w - \hat{w}\|$, and in later iterates ends up "bouncing around" a little, most likely due to the highly discontinuous re-sorting operation involved. In going from one iterate to the next the estimate may become better in some portions of the domain of the function, and worse in others. Nevertheless the iteration is useful as it generally improves initial estimates and these later oscillations are relatively smaller than the initial steps of more substantial improvement. For this particular system and simulation data, the first iterate tends to be quite inaccurate, the second iteration makes a large improvement and suddenly the estimate looks reasonable, and then after this subsequent iterates make smaller changes and, roughly, converges to a fixed point. The larger the noise level, the farther out into the sequence the iterates are still producing substantial changes.

Next let's explore the how the choice of the weights Γ_w, Γ_z, and Γ_e in the problem formulation affects results. Without loss of generality we can fix one and vary the other two, so we choose to let $\Gamma_w = 1$. We are now using the first 800 points of the $\sigma_e = 0.2$ data, and taking the eighth bootstrap iterate as the final estimate. In Figure 4.7 we fix $\Gamma_e = 1$ and show results for Γ_z chosen from $\{0, 1, 5, 10, 20\}$. Γ_z is having a modest effect here, producing similar results for the values tried: 0, 1, 5, and 10. The nicest-looking estimates

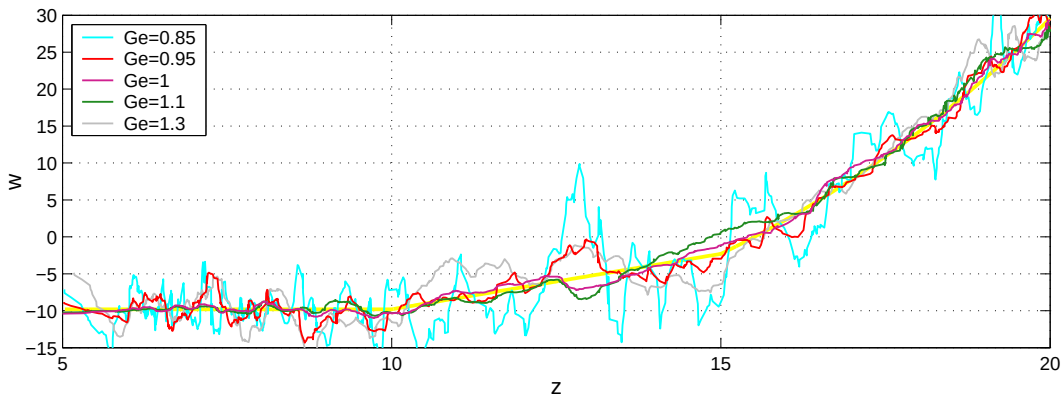


Figure 4.8: Scatter plots of estimates using various Γ_e.

correspond to the lower values. For $\Gamma_z = 20$ the estimates start to degrade significantly. Note that $\mathcal{J}$ with $\Gamma_z = 0$ is $\mathcal{J}_w$.

Next fix $\Gamma_z =$ and look at the effect of Γ_e. Figure 4.8 shows estimates for several different Γ_e chosen from $\{0.85, 0.95, 1, 1.1, 1.3\}$. Here the best estimates are with $\Gamma_e = 1$ or $\Gamma_e = 1.1$. In our experience this knob is a good way to control certain aspects of the estimation process. As a very general rule of thumb, as the size of the noise estimate is constrained more tightly, the estimates are fuzzier, but also less biased. Notice the estimate with the smallest value of Γ_e is, well, noisy. Presumably this is because we have limited the size of e, hence in our estimates we have to look to w to explain any noise in the output y, hence w has more of the characteristic of noise. Conversely, as the noise constraint is relaxed the scatter plot becomes smoother (but more biased). Now the estimation process can use more e to explain the output, thus allowing it to find (z, w) pairs that make the dispersion smaller. If this is the case, then the estimated noise will start to become more correlated with y and other signals in the problem than we expect, statistically speaking. In Figure 4.9 we show the autocorrelation $\rho_e(h)$ of $\hat{e}$ for lags 0 through 200, along with its correlations with $\bar{y}$, $\bar{u}$, $\hat{w}$, and $\hat{z}$, and 95% confidence bounds (statistics says 95% of the lags should lie within these bounds). We see that for the looser noise constraints the noise estimate becomes highly correlated with $\bar{y}$, $\bar{w}$, and $\bar{z}$.

Often our experience has shown that a good value of Γ_e is near one, that is to say, it is often reasonable to constrain the size of the estimated noise to be less than the variance of the process. Sometimes we prefer slightly smaller values, say 0.95, as we have found this to be able to produce estimates with less bias. However this choice depends

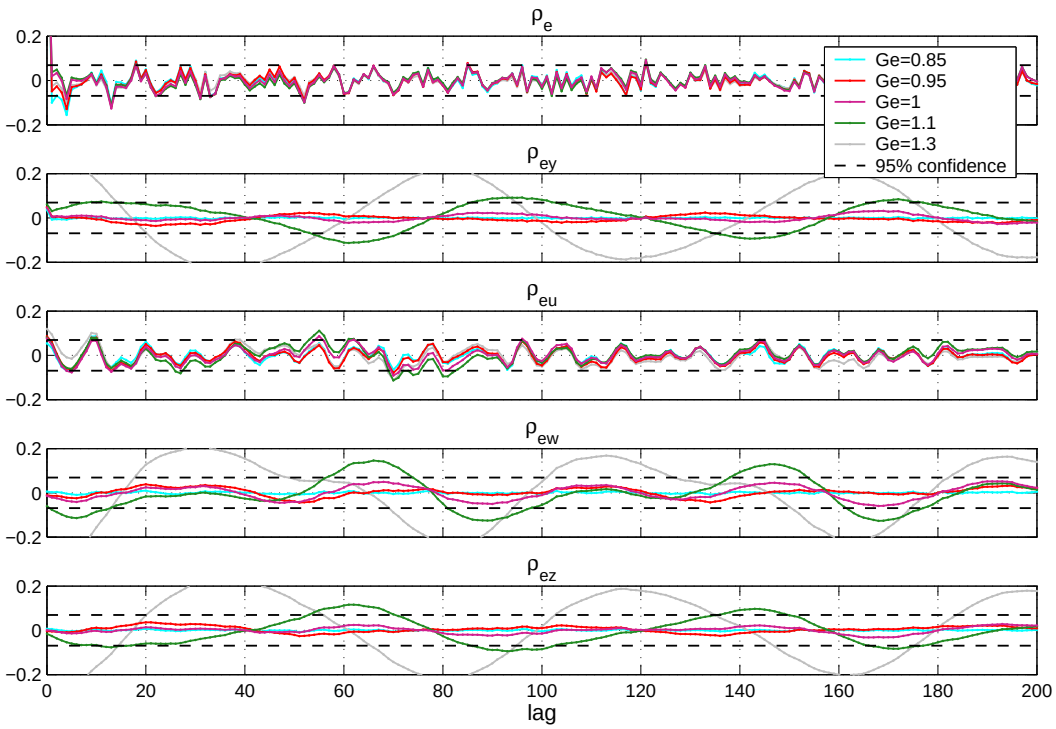


Figure 4.9: Correlation functions of the estimates of Figure 4.8, for various Γ_e.

on having the variance σ_e of the noise process correct. Otherwise it becomes an iterative process of trying a value, inspecting the estimates for clues as to its appropriateness, and modifying the guess. In practice, when we are estimating a function that is not known, we don't have the luxury of trying different values of σ_e and Γ_e until some combination produces estimates that match the true nonlinearity. In this example we see that the wrong noise constraint, particularly one that is too loose, can produce scatter plot estimates that look reasonable and smooth, yet are more biased than estimates produced with perhaps more accurate knowledge of the noise variance level. This is why it can be important to check the correlation of the estimated noise with other signals in the problem.

It is interesting to compare estimates made using a simpler objective criteria, and ones using more traditional approaches to this type of problem, in order to see that the dispersion objective is of some value. Perhaps these situations will give us an idea how hard or easy this particular estimation problem is, and provide a reference with which to judge the dispersion estimates.

First we consider replacing the objective of the optimization formulation, $\mathcal{J}(z, w)$,

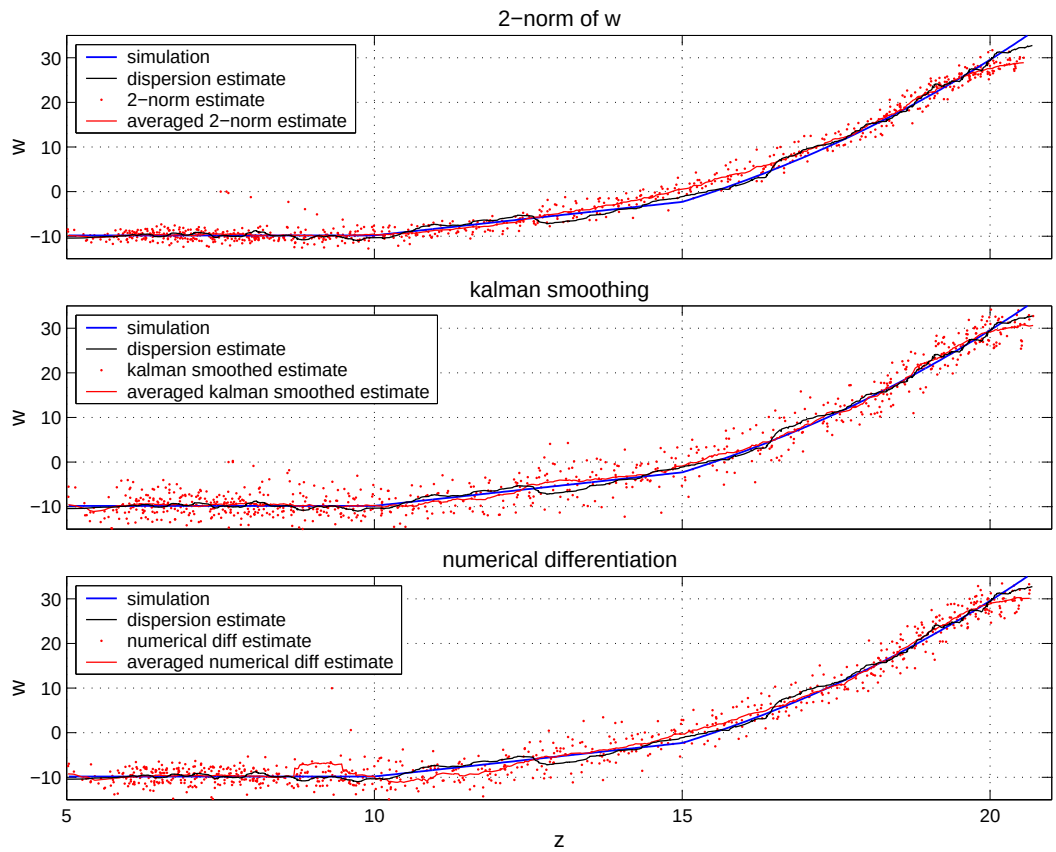


Figure 4.10: Alternative ways to compute estimates.

with the simpler criteria, $\|w\|^2$, the 2-norm of w. The formulation thus becomes

$$f = \begin{array}{ll} \underset{f}{\text{argmin}} & \|w^f\|_2 \\ \text{subject to} & \frac{1}{L}\|e^f\|_2^2 \leq \Gamma_e\, \sigma_e^2. \end{array} \tag{4.7}$$

Here the problem does not depend on the parametrization of the feasible z at all. The estimation problem decouples into two steps of first estimating e and w, and then computing $\hat{z}$ as the signal compatible with these estimates. Since the constraints and criteria do not depend on z in this formulation, the optimization problem is immediately convex and quadratic in the decision variables, and no bootstrapping is necessary. As in previous examples we use the first 800 points of the $\sigma_e = 0.2$ data set, with $\Gamma_e = 1$. The resulting scatter plot is shown in the top panel of Figure 4.10. For purposes of comparison we also show the previously-computed dispersion estimate. Portions of the estimated signals are graphed in Figure 4.11, along with the dispersion estimates. These two figures also

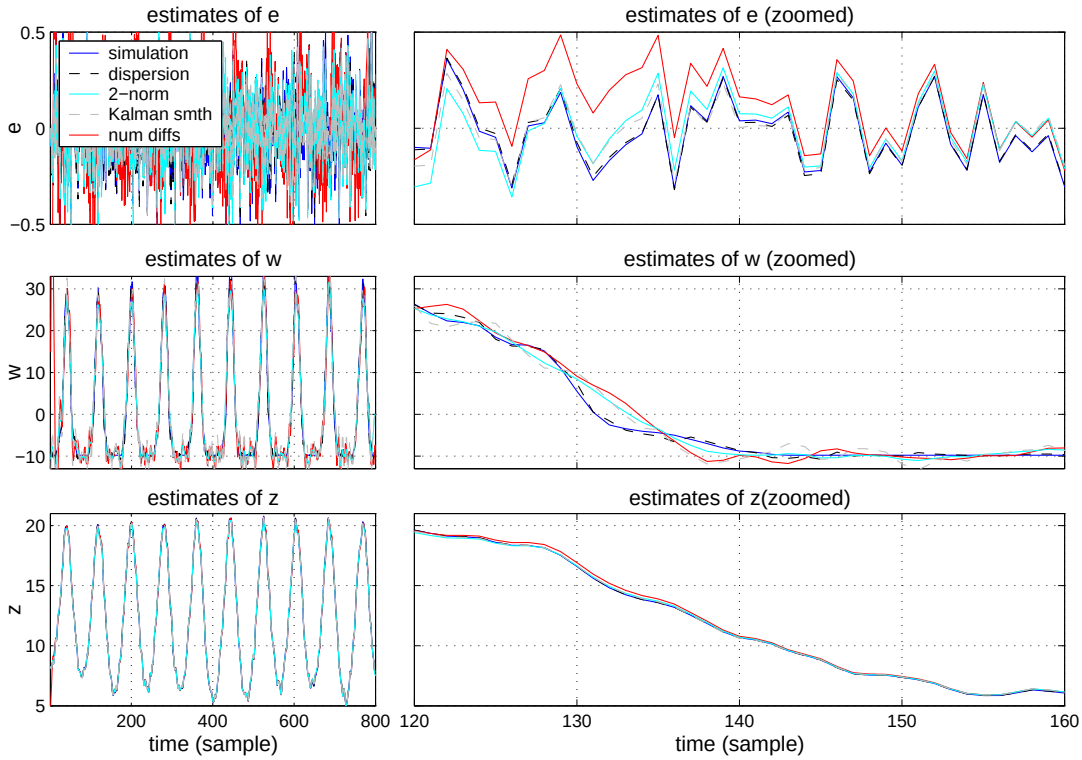


Figure 4.11: Alternative ways to compute estimates–signals.

show 2 other types of estimates to be explained below. We see that the results are again reasonable, in that the trends are certainly right and the estimates appear unbiased. But the variation of the estimated signal about the true, simulation values is much larger than with the dispersion measure. Presumably this is because the dispersion criteria penalizes this jumpiness, and enforces smoothness. When we check, the 2-norm of these estimates are indeed (slightly) smaller than those of the dispersion estimate; however their dispersion measures are much larger. In spite of the higher variance, on average the estimate is roughly centered about the true nonlinearity, general trends are correct, and we can certainly learn a lot about the function from this estimate. The fourth thing plotted in the top panel is an averaged version of the 2-norm estimate; each point in this plot is an average over a local window of the 2-norm estimate, the window size being chosen so that they contain approximately 20 points. This filtered version of the 2-norm estimate is now comparable to the dispersion estimate, and beats it in some regions of the z-domain. Of course this involves an extra step, and a new decision about how much averaging is needed to reduce noise yet retain the individual features of the function.

Another way to form estimates is to use Kalman smoothing (KS). Remember that most fundamentally we are solving an estimation problem involving unknown signals, a known linear plant, and knowledge that some of the signals are related by a static operator. Given a plant, its output, and partial knowledge of inputs, the task is to estimate the remaining inputs. A common device used to solve this type of problem is Kalman filtering, or in the offline setting, Kalman smoothing. Thus we may want to compare results with KS estimates. It is an optimal solution to the problem of estimating e and w (or equivalently, estimating the system state) when these are known to be independent white Gaussian signals of known variance. And in practice it is often used when these assumptions are not strictly met, as it often produces useful results. In the present situation we have the system

$$y = \mathcal{L}_y \begin{pmatrix} u \\ e \\ w \end{pmatrix}, \tag{4.8}$$

and the job of the KS is to produce estimates $\hat{e}$ and $\hat{w}$ of e and w given $\mathcal{L}_y$, input-output data, and assumed variances for e and w. To produce the estimate $\hat{z}$ of z we can then use

$$\hat{z} = \mathcal{L}_z \begin{pmatrix} u \\ \hat{e} \\ \hat{w} \end{pmatrix}. \tag{4.9}$$

Taking advantage of the fact that we know the correct variance of the realizations of e and w for our simulation data record, we computed the KS estimates in this way. The scatter plot of $(\hat{z}, \hat{w})$ is shown in the second panel of Figure 4.10, once again accompanied by an averaged version, and the dispersion estimate.

The results are similar to the 2-norm estimate (but with higher variance). This is not surprising. What the KS is actually doing is computing the solution of

$$(\hat{e}, \hat{w}) = \underset{e,w}{\operatorname{argmin}} \|e + \gamma w\|_2 \quad \text{subject to} \quad y = L_y \begin{pmatrix} u \\ e \\ w \end{pmatrix}, \tag{4.10}$$

with γ chosen according to the relative size of the variances specified for e and w. It finds those signals with smallest 2-norm which explain the input-output data (note that specifying the input signals is more or less equivalent to specifying a state trajectory). In the stochastic setting this is the optimal thing to do because such signals occur with higher

probability as the realization of a Gaussian white noise process. The contribution of KS, besides being a computationally efficient iterative algorithm, is to quantify the optimal tradeoff between trying to optimize both $\|e\|_2$ and $\|w\|_2$ based on their known variances, and in the absence of other information about the problem—in this case, neglecting that the relationship between z and w is static.

Now the family of estimates produced by (4.10) as γ varies is the same as that produced by solving

$$(\hat{e}, \hat{w}) = \underset{e,w}{\operatorname{argmin}} \|w\|_2 \quad \text{subject to} \quad y = L_y \begin{pmatrix} u \\ e \\ w \end{pmatrix}, \ \|e\|_2 \leq \alpha \tag{4.11}$$

as α varies; for every γ, there is an α such that the estimates produced by (4.10) and (4.11) are the same (Wemhoff 1998). The 2-norm-of-w formulation whose estimate is shown in Figure 4.10 is equivalent to (4.11), for a specific choice of α. This, then, is why the Kalman smoother estimate and the 2-norm-of-w estimates look similar.

Another interpretation of this class of problems is that of computing an ill-posed inversion of a linear system, namely we are inverting L_y to estimate (e, w) from knowledge of y. Here the solution is nonunique, and in these situations a common practice is to add *regularizing* constraints and criteria to make the problem well-posed. A common regularization choice is a norm or smoothness consideration on the variables being solved for. In this case minimizing $\|w\|_2$ while constraining $\|e\|_2 \leq \alpha$ leads to the 2-norm-of-w estimate, whereas minimizing $\|w + \gamma e\|_2$ leads to the KS estimate.

In addition to using KS as an alternative to the dispersion formulation for finding estimates, we can also compare results with a more ad-hoc approach. Looking at the problem formulation again we see that w actually represents the acceleration of the mass. Since we have a position measurement, why not 1) use that to compute the stretch of the cord i.e. z and 2) differentiate twice to compute the acceleration of the mass i.e. w. We call this the "numerical differencing" estimate. Since the position measurement is noisy, and this noise is amplified by the numerical differentiation, to give this a fair shot the first thing we do is to filter higher frequency noise out of the position measurement. We used a 3rd-order butterworth with cutoff at 5% of the sample frequency, which seems to produce about the best results from among several choices of order and bandwidth. The scatter plot of the estimates produced in this way is shown in the third panel of Figure 4.10, again with

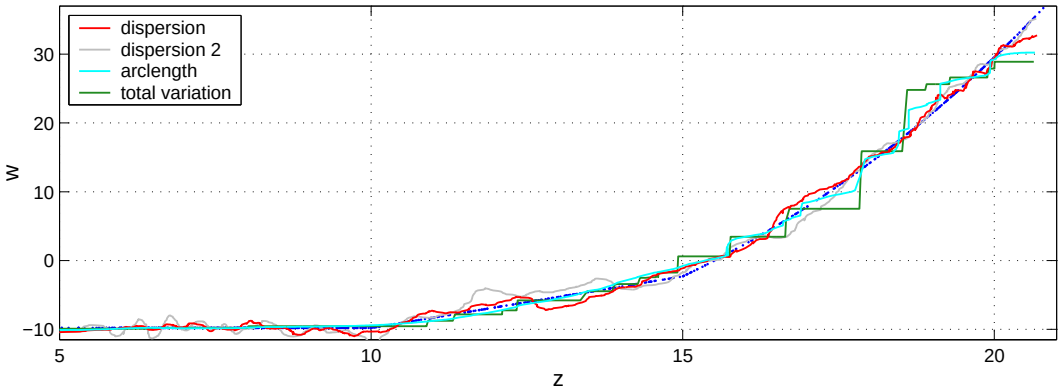


Figure 4.12: Estimation formulations using various staticness measures.

an averaged version and dispersion estimate. Results are similar to the 2-norm-of-w and KS estimates, in that the estimated scatter plot has a high variance, and the averaged plot looks pretty good.

Finally, with the same data set we compute estimates where the dispersion objective is replaced with some of the more dispersion-like objectives mentioned in Table 4.1. Figure 4.12 shows a portion of the scatter plot estimated using the second-difference dispersion $\mathcal{J}_2$, arclength $\mathcal{J}_{\mathrm{A}}$, and total variation $\mathcal{J}_{\mathrm{TV}}$, along with the usual dispersion estimate. Figure 4.12: disp1, disp2, arc, tv Each of these four has a somewhat different characteristic than the others. The second-difference dispersion estimate meanders around more than the first-difference dispersion; this is especially noticeable in the flat portion of the function between $z = 5$ and $z = 10$ in this plot. The arclength and total variation, and more so with the latter, have regions of smooth, flat estimates, with occasional jumps from one level to another, producing a staircase effect. These estimates look especially good in the flatter regions of the nonlinearity, as compared to the dispersion estimates. In general the arclength estimate is preferable to the total variation estimate. Of the four, the arclength estimate is the best for smaller z where the function is flatter, and the first- or second-difference dispersion estimates are better for larger z, where the function has a larger slope. In the bottom panel of Figure 4.4 we plotted several arclength estimates, using different data lengths, in order to compare to the dispersion estimates in the top panel. The same general trends are evident here — estimates do not necessarily improve when using longer data records, and the arclength estimates are more accurate for smaller z.

In summary the main estimation formulation we have used is to minimize disper-

sion, while constraining the norm of the noise estimate. As z is not measured, we use the bootstrap idea to go about solving this formulation. This approach produces reasonably accurate estimates. One shortcoming we have found is that estimates do not necessarily improve when more data is used. Another is that estimated scatter plots, although less noisy than those produced by alternative methods such as Kalman smoothing, tend to be biased, and can be misleading in terms of missing smaller details in the graph of the functions being estimated. At this point it is not clear whether these shortcomings are associated with the formulation, the solution method, or both. In this example the bootstrap method finds estimates with a small dispersion, but they are not optimal; we saw that the simulation signals are feasible, and have smaller dispersion. Perhaps a solution method that does a better job of solving the optimization problem would produce better estimates.

At this point we could try more complex combinations of objectives and criteria. For instance perhaps an objective that is a weighted sum of dispersion and $\|w\|_2$ would produce estimates with the smoothness of the dispersion estimates, but less bias as is characteristic of the 2-norm-of-w estimates. As another example, we have found for this example that starting off with 3 bootstrap iterates using the dispersion objective, followed by more iterations that use the second-difference dispersion objective, produces estimates that are slightly better. Or we might try adding constraints on the correlation of e with other signals in the problem. However, in general we find that the changes these variations produce are small, and sometimes beneficial but not always. Perhaps more importantly, from a usability standpoint the more complex the criteria and algorithm gets, the harder they are to use, to get to work. In real problems we wouldn't have the luxury of being able to compare estimates with the true quantities to figure out which combination of elements to use, and so it would be hard to know when to stop tweaking. The basic dispersion + noise constraint + bootstrap approach is simple, appears to be fairly robust, and produces reasonable estimates.

4.3 Example: ECP

For more estimation experience we consider another example. At UC Berkeley we have in one of the educational labs a commercial product by ECP Systems (`www.ecpsystems.com`), their model 220 Industrial Emulator. This is sold as a tool for education in controls, providing a testbed for experimentation in control and system identification. It consists of a

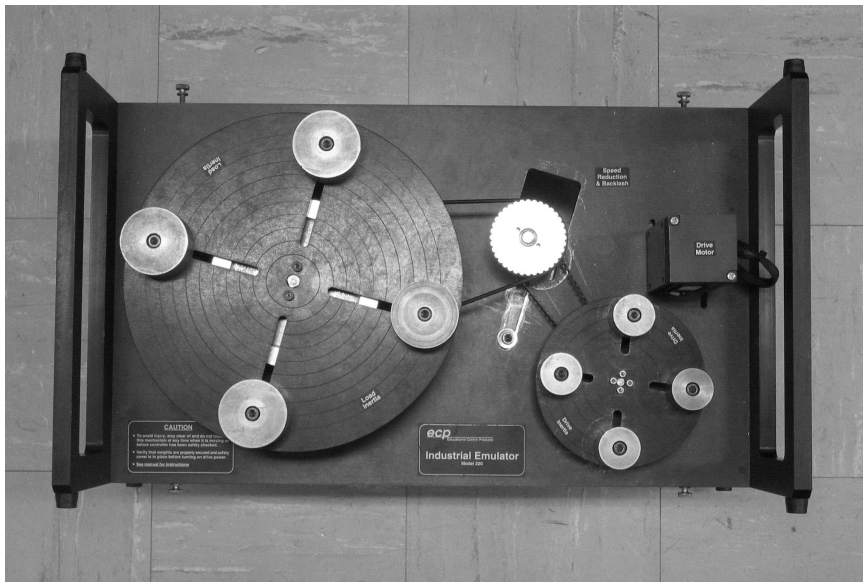


Figure 4.13: ECP "Industrial Emulator" controls experiment.

drive motor and a load inertia which can be coupled to it with a flexible band. A picture of such a device Figure 4.13. The product features a realtime onboard dsp controller and software to interface with a PC.

The reason we are interested in this system is because: 1) it provides realistic numbers for an example and 2) several static nonlinearities in the system can be considered. These include nonlinear friction acting on both the drive and load systems, an input that can saturate, as well as a backlash nonlinearity that can be introduced into the idler pulley shaft by loosening a set screw. It is also a relatively nice controlled experimental setup from which we can obtain input-output data if desired.

The motor is connected to a smaller "drive" inertia which is in turn connected to an idler pulley by inelastic belts, whose shaft is connected to the larger load inertia by the flexible belt. We can simplify things by grouping components connected by rigid belts into the "drive" and "load" parts of the system. The device has two degrees of freedom can be nominally modeled as two second-order, double integrator systems connected by a spring, namely the drive and load systems. The configuration of the system is defined by the shaft angles of the drive and load, θ_1 and θ_2 respectively (measured in radians).

When the load inertia is disconnected, i.e. the elastic band is removed, the system reduces to a second-order double integrator system, with viscous and nonlinear damping.

J_*	0.0025	J_2	0.032
c_*	0.003	c_2	0.003
$k_t k_a$	0.23167	k_L	320
k_c	10/32767	r_1	0.01
k_e	$16000/(2\pi)$	r_2	0.06
k_s	32	r_{p_1}	0.02
h	0.010608 sec	r_{p_2}	0.03
k_P	0.1		
k_D	0.2		

Table 4.4: Constants in the ECP model.

We start with this simplified setup. The equation of motion we use is

$$\ddot{\theta}_1 J_* + \dot{\theta}_1 c_* + \Psi_*(\dot{\theta}_1) = T_D \tag{4.12}$$

or in state-space form

$$\frac{d}{dt}\begin{pmatrix}\theta_1\\ \dot{\theta}_1\end{pmatrix} = \underbrace{\begin{bmatrix}0 & 1\\ 0 & -\frac{1}{J_*}c_*\end{bmatrix}}_{A_c}\begin{pmatrix}\theta_1\\ \dot{\theta}_1\end{pmatrix} + \underbrace{\begin{bmatrix}0\\ \frac{1}{J_*}k_t k_a k_c\end{bmatrix}}_{B_c} u - \underbrace{\begin{bmatrix}0\\ \frac{1}{J_*}\end{bmatrix}}_{B_{w,c}} \Psi_*(\dot{\theta}_1), \tag{4.13}$$

where J_* is the sum of the inertias of the components in the drive part, reflected onto the drive axis, c_* is a similar sum of the viscous damping in the drive section, and $\Psi_*(\dot{\theta}_1)$ represents a nonlinear friction term that is a function of velocity (an idealized friction model). The actuator is a DC servomotor that can be modeled as a torque source due to an internal servo circuit (5 kHz bandwidth for internal current loop). The motor produces a torque T_D, which is related to the output u of the controller software as $T_D = k_t k_a k_c u$, where these motor constants are provided to us. The inertia and damping parameters are known approximately.

We consider first a simulation study of the system. See Table 4.4 for values of the ECP constants that were used for this. The task is to identify the nonlinear friction characteristic, given the linear model (i.e. given the parameters J_*, c_*, and the motor constants). Since the methods we are using depend on having a discrete time model of the linear plant, the first thing to do is discretize (4.12). We choose a simple Euler-type discretization, with state defined in terms of the continuous time states as

$$x_k := \begin{pmatrix}\theta_1(kh)\\ \dot{\theta}_1(kh)\end{pmatrix} \tag{4.14}$$

where h is the sampling time, and using the differencing approximation

$$\frac{x_{k+1} - x_k}{h} \approx \frac{d}{dt}\begin{pmatrix}\theta_1(kh)\\ \dot{\theta}_1(kh)\end{pmatrix} \tag{4.15}$$

we have

$$x_{k+1} = (I + hA_c)x_k + hB_{u,c}u_k + hB_{w,c}\Psi_*(x_{2,k}). \tag{4.16}$$

Since the system is only marginally stable a controller is applied, in discrete time, using a simple proportional with inner-loop feedback law

$$u_k = k_s k_P(r_k - k_e\theta_1(kh)) - k_s k_D(k_e\theta_1(kh) - k_e\theta_1((k-1)h) \tag{4.17}$$

This requires an additional state and so we augment the state to be

$$x_k := \begin{pmatrix}\theta_1(kh)\\ \dot{\theta}_1(kh)\\ \dot{\theta}_1((k-1)h)\end{pmatrix}. \tag{4.18}$$

Here k_P and k_D are the proportional and derivative gains, and r_k is the reference signal at time k. The other constants are added by the ECP software and we include them here so that the units of signals are consistent with the ones used by that product. We can define feedback matrices K_x and K_r according to

$$u_k = K_x x_k + K_r r_k. \tag{4.19}$$

Applying this feedback we have finally the discrete equations of motion

$$x_{k+1} = \underbrace{(I + hA_c + hB_{u,c}K_x)}_{A} x_k + \underbrace{hB_{u,c}X_r}_{B_u} r_k + \underbrace{hB_{w,c}}_{B_w} \Psi_*(x_{2,k}) + \underbrace{\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}}_{B_e} e_k, \tag{4.20}$$

where now we have added a possible disturbance. The output of the system is

$$y_k = k_e x_{1,k} + \begin{bmatrix}1 & 0\end{bmatrix} e_k. \tag{4.21}$$

Here e_k is composed of two scalar signals, a measurement and a disturbance noise.

The part of this system that is of interest to us is the static friction nonlinearity Ψ_*. In keeping with the notation of previous sections we define $z_k := x_{2,k}$, the input to Ψ_*,

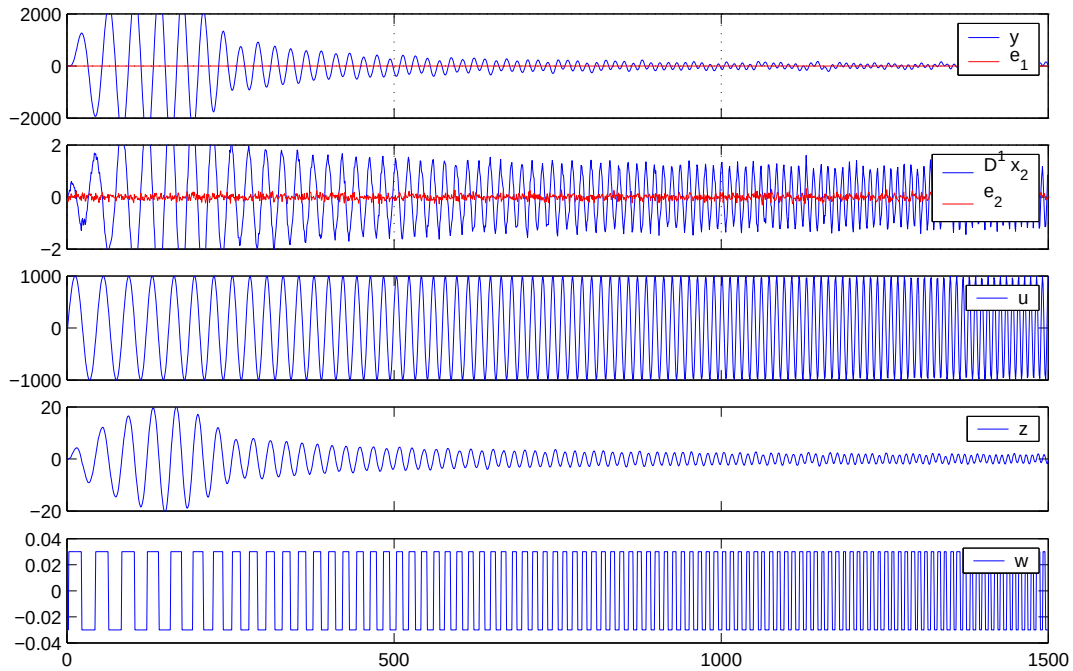


Figure 4.14: Simulation data for ECP noload model.

and $w_k := \Psi_*(z_k)$, the output. It is these signals, along with e_k, that we need to identify. Simulation data is generated using a Coulomb friction model,

$$\Psi_*(z) := \begin{cases} -.03 & z < 0 \\ .03 & z \geq 0 \end{cases} \tag{4.22}$$

a chirp reference input signal to the controller (which we hereafter call u in order to be consistent with our usual notation), uniform white noise for e_1, and Gaussian white noise for e_2. See Figure 4.14 for a plot of a typical simulation. A standard deviation of 0.35 is used for the measurement noise (which is tiny compared to the units of y), and a standard deviation of either 0.1 or 0.02 for the disturbance noise, which is substantially larger relative to signals it is added onto. Regarding this, perhaps the correct setting in which to judge the size of the disturbance noise is to compare its size to the sum of the signals that add in to determine the increment of x_2 from one sample to the next. The first difference of x_2 is plotted in the second panel of the figure, along with e_2 to get an idea of their sizes.

Using simulated input-output data we now look to identify unknown signals, and the nonlinear friction characteristic. In this system there is a very accurate measurement of θ_1 made with an optical encoder. The z signal is the derivative (or difference in discrete

time) of this. So although z is not, strictly speaking, measured, it is *almost*; a slightly noisy version of it can be derived from the signals we have available. This is reflected in the bootstrap iteration in that the first iteration produces most of the improvement, and later iterates cause only very small changes. The second iterate is a very slight improvement on the first, and then it's essentially done. This is because in each iteration the program is quite constrained, by the linear system and input-output data, as to how much it may vary z from the particular solution z^0. Therefore z^f, and the sort operator, change very little from one iteration to the next, and so successive iterations are solving nearly identical problems. The iteration, along with some of the problems mentioned in the previous section, is, in a sense, bypassed in this example. For this reason we will use the product of the first or second iterate as the "solution" of the formulation.

The only formulation we consider here uses the dispersion function as objective while constraining the size of $\|e\|_2$, which is by now our standard approach. Figure 4.15 gives an idea of how well and how easily it works, and we attempt to explore how the choice of design weights affects the estimate. Unless otherwise specified the estimates are made from the data in which $\sigma_{e_2} = 0.1$. As there are now two noise signals we have to choose two noise constraints $\Gamma_{e,1}$ and $\Gamma_{e,2}$, along with Γ_w and Γ_z for the dispersion. For the latter two we use 1 and 0.05, respectively, as we found that these allow us to find good estimates. This choice is motivated by the fact that the set of values that z ranges over is much larger than w in this problem due to the choice of units, therefore a small weight is used to make the $\|\Gamma_z D^1 P^z z\|_2$ part of the dispersion on a comparable level as the $\|\Gamma_z D^1 P^z w\|_2$ part (recall from Section 3.3.1 that dispersion is not invariant to scaling). A good rule of thumb we have found is that the two pieces should play a balanced role. In this case when we blindly set $\Gamma_z = 1$, estimates are poor.

With this choice and a fixed portion of the data set, in the top panel of Figure 4.15 are the estimates using different constraints on the disturbance noise. As usual, as the constraint gets tighter the scatter plot estimates get "noiser", because $\hat{z}$ and $\hat{w}$ are being used to explain variance that $\hat{e}$ no longer can. To our eye the estimate with $\Gamma_{e,2} = .55$ looks best. As usual the rule of thumb here, that one decrease the noise constraint until estimates begin to look noisy, seems to work. The second panel has two estimates each (using different disturbance noise constraints) for two different portions of the data record, namely the first 500 points, or the next 500 points. The thing to note is that the region of the domain of the friction characteristic that is "identified" is much smaller in the latter

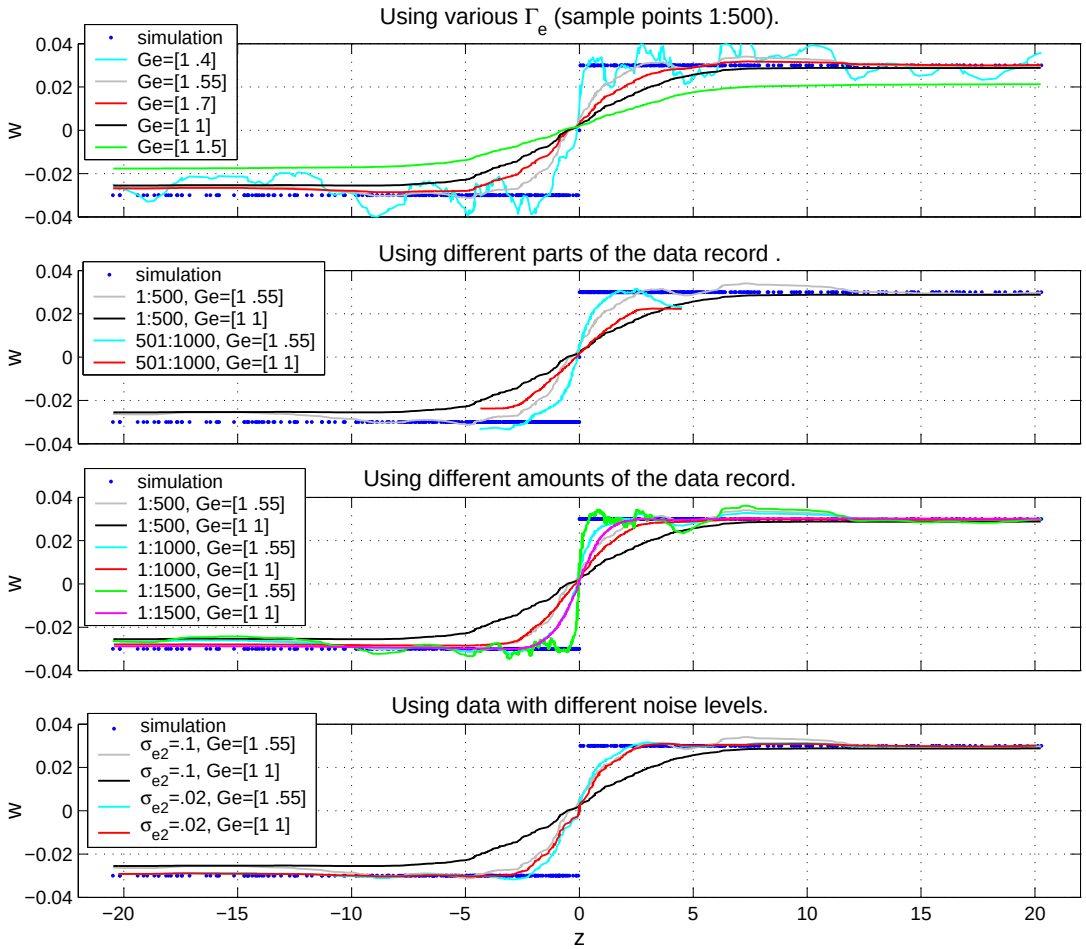


Figure 4.15: Trials for estimation of simulated Coulomb friction nonlinearity in ECP noload model.

case, simply because the oscillation of the system is slower in this part of the record. In the third panel we show two estimates each for 3 *lengths* of data record. Unlike the bungee example, in this example estimates get distinctly better when more of the data is used. Lastly in the fourth panel we show that if the data has less noise, then estimates will be better (in this case the size of the disturbance noise is changed).

Now we move on to a more worthy challenge, where the operator to be estimated is not already known to us! The problem is the same as the one just considered except that instead of using simulation data we will endeavor to use input-output data measured on the ECP experiment. We used the experiment without the load inertia attached and the aim is to identify a velocity-to-friction map, and so we use the model (4.20) and (4.21), with the same parameters as in Table 4.4. In fact, since our identification method depends to

some degree on having an accurate linear model, the first thing we did was to run a linear identification procedure to estimate J_* and c_* (this is where the values in that table come from). Although we are thinking the system is nonlinear, we were still able to find what we think are good numbers for these two parameters.

Next we sampled responses of the system. The ECP software has a limited number of reference trajectories available; we chose to use chirp signals, linear in frequency (the same input that was used in the simulation just now). Figure 4.16 shows the output, y, for nine such experiments. The x-axis here uses units of samples (recall the sample time is $T = 0.010608$ seconds). We observed the reference input and encoder output signals as we varied amplitude, and frequency between 0.02 and 8 Hertz. Using this data, the identified linear model, and the dispersion/noise-size formulation, we set off computing estimates.

Guided by our experience with the simulation model we fixed $\Gamma_w = 1$ and suspect $\Gamma_z = 0.05$ is a good choice, and sought to locate appropriate noise constraints. However in this situation we don't know the noise variance σ_e so conveniently as before. We have some idea for the measurement noise; the optical encoder should never be off by more than plus or minus half a count. If we think this noise process is reasonably well modeled by a uniform distribution, $e_1 \sim U[-0.5, 0.5]$, then it has variance $\sigma_e^2 = 1/12$, or $\sigma_e = 0.29$. The disturbance size is harder to judge. The disturbance quantity is meant to account for a lot of things; external disturbances, unmodeled dynamics, etc, and its nature is not as clear as the measurement noise. What we can say is that without a disturbance noise channel in the model we would often ran into infeasibility, where the measured input-output data is not consistent with the linear model we are assuming. It seems that some amount of disturbance is necessary to explain the data.

We take the slightly different point of view in this problem of fixing $\Gamma_e = [11]$, and searching for the "correct" σ_e; they affect the same thing in the same way, it's just how we think about it. We found that the choice σ_e that results in the best estimate (or what we think is best, since this time we don't actually know) depends strongly on the data record used for estimation. The way we decide an estimate is probably good or not is based on the intuition built up with simulation examples. We often see that the thing to do is lower the noise constraint until scatter plots start to get fuzzy, and then we're close to the best estimate that the dispersion formulation will find.

Initially it was a hard task to find the right σ_e, especially given the fact that it's different for different data records. Eventually we recognized that it's the value of $\sigma_{e,2}$ that

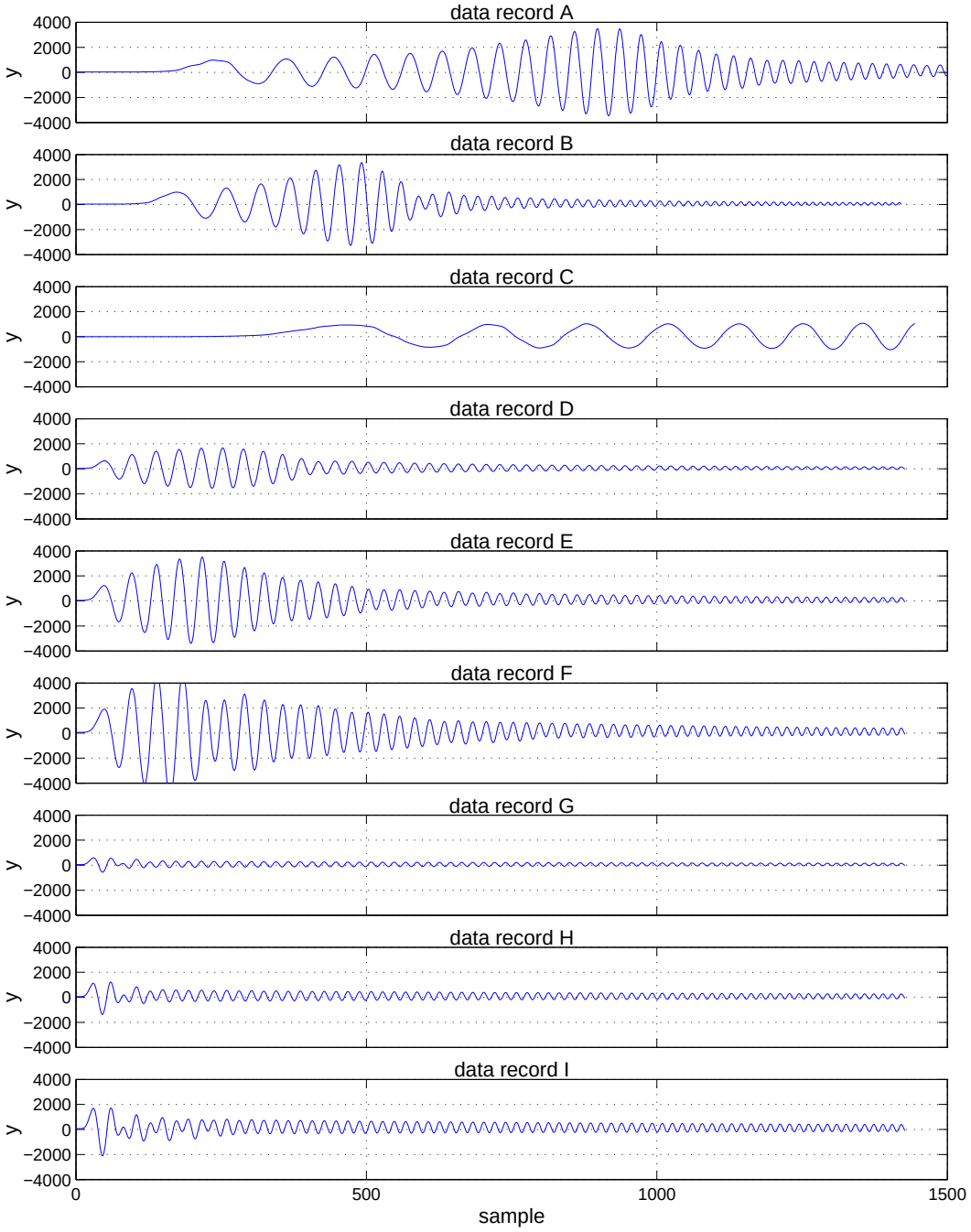


Figure 4.16: Excitation of ECP drive system during 9 chirp experiments.

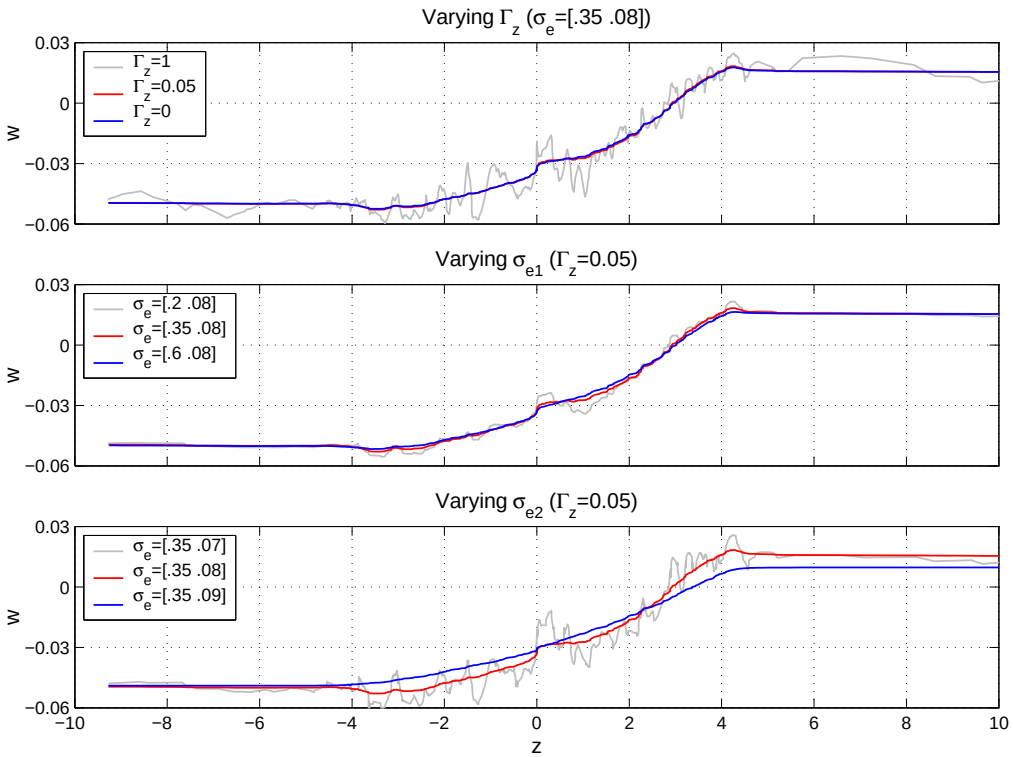


Figure 4.17: Varying Γ_z and σ_e.

needs to change, while the value of $\sigma_{e,1}$ we eventually settled on, which is 0.35, works for most data records. In a way this makes sense. Our understanding of the measurement noise is quite good and we expect it to essentially not depend on the experiment, but our understanding of the process noise in this example is not. Here, the level of process noise appears to depend on the experiment performed.

In Figure 4.17 we consider the weighting choice, using data set H in this case. Through trial and error we settled on a value of $\sigma_e = [.35\ \ .08]$ as likely being a good constraint level, and in this figure the constraint level used in the formulation is perturbed about this point to see its effect. We see that when the level of one is raised estimates becomes more smooth, and when made tighter, more noisy. In this picture we also note that the estimate appears to be more sensitive to the choice of $\sigma_e(2)$ than $\sigma_e(1)$.

It turns out that for these estimates, the correlations of the estimated e_1 with other signals changes when its constraint level is varied; for the choice $\sigma_{e,1} = 0.6$ there is significant travel outside the 95% confidence bounds, and for values less than this, and for all values of $\sigma_{e,2}$, these correlations satisfy white noise assumptions. It seems we could use

Estimate	Data Record	samples used
1	H	1:500
2	E	501:1000
3	I	1:500
4	I	501:1000
5	A	1:700
6	E	1:500
7	F	1:500

Table 4.5: Data records of estimates in Figure 4.18

this fact to alert us to good choices of constraint for e_1. On the other hand, correlations associated with e_2 estimates are outside the confidence bounds for all five choices, and they do not seem to be sensitive to this choice. However we do not predictably see this feature in all the examples we've tried, and are still trying to understand how to best make use of correlation information.

We found that between data records with amplitude and frequency ranges that are comparable the "best" scatter plots, although certainly not the same, have a consistent shape. On the other hand, scatter plots estimated using data records with different natures varied more substantially. In the top panel of Figure 4.18 we show the most appealing estimates for 7 portions of data records. Table 4.5 matches up each estimate with the data record it is based on. Estimates 1, 2, 3, and 4, which look similar, came from similar-looking data records: frequencies around 4 to 5 Hz, and lower amplitudes. Estimates 5, 6, and 7 belong to portions of data records with slower frequencies and higher amplitudes, and these estimates are quite different than the first four.

To convince ourselves that these are indeed reasonable estimates of something and not just figments of our optimization, we ask whether they can be used to improve the capability of the linear model to match the response of the ECP system. To proceed we tried matching piecewise linear functions to the scatter plots, folding these into the linear model, and computing the response of this augmented model to some of the input data from the ECP experiments, for which we also have records of the actual ECP output.

Choosing Estimate 1 in the first panel of Figure 4.18, we have drawn in such a fit. The bottom three panels correspond to three of the ECP data records, and contain the response of the linear model to the input signal of that record, the response of the model that is augmented with the piecewise linear fitting function, and the output of the ECP itself. What we see is that for the 9 data records on the whole it looks like the added friction

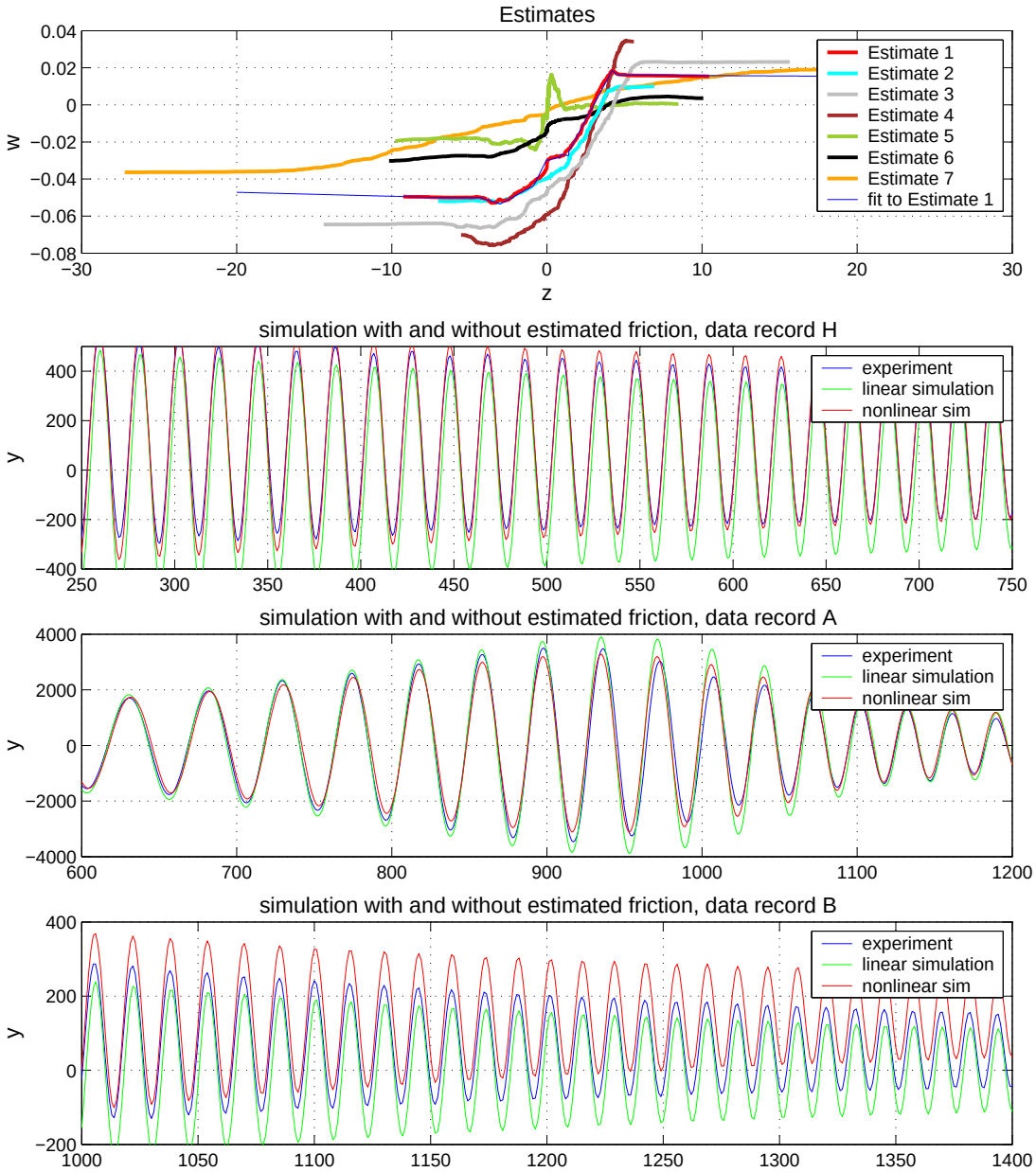


Figure 4.18: Estimates of Ψ_*, a piecewise linear fit, and simulations.

model is a slight improvement. The simulation for data record H, which is the data that was used to compute Estimate 1, is a clear improvement over the linear model. The second example is perhaps a marginal improvement and the third is questionable, but on the whole the estimate does seem to have value.

One thing to note is that the estimates, at least the first four, do not possess the shape that we expect a friction characteristic to have. For one thing note that most of the estimates do not go through zero, and instead for zero velocity there is a nonzero frictional force being exerted! This is most likely a misinterpretation of whatever phenomenon in the system that this characteristic is capturing; certainly there is some amount of Coulomb-like friction present, but it seems likely there is something else going on. Perhaps it is relevant to mention that while collecting this data we did notice some preference for a direction in which the drive inertia would turn most easily (with the power off); we're not sure what might be causing this or how best to model it. In those simulations using the augmented model shown in Figure 4.18, for data records H and B there is a propensity for the nonlinear model to maintain a nonzero mean which enables it to match the output data better than the linear model.

Another thing to mention is that viscous friction shows up in the model in the same way that the nonlinear friction characteristic does, namely as a static mapping between velocity and force. In fact our first estimate of the viscous damping c_* from linear estimation methods was not quite accurate, and this residual mismodeling could be seen in some of the estimate of Ψ_*, in that the outer tails had a nonzero slope. When we adjusted the viscous damping coefficient of the linear model accordingly, this disappeared from the estimated scatter plots.

In addition to adding the disturbance noise channel, it was at times necessary to allow unknown initial conditions to be estimated, as opposed to being fixed, in order to make the optimization problem feasible This occurred especially when using portions of data records such as 501:1000, ones where the initial conditions are not zero.

We now move back to working with simulation data for the ECP model. This time we consider the full fourth-order model, in which the elastic connects the drive and inertia loads. This allows us to introduce two additional meaningful SISO nonlinearities into the problem. The first is a nonlinear friction term for the load inertia, in addition to the one for the drive. Another arises when a backlash is introduced into the idler pulleys; this manifests in the elastic band having a deadband characteristic, where there is some

amount of stretch possible in the band without the drive inertia feeling a reaction moment.

In this case the system equations are, analogously to (4.13),

$$\frac{d}{dt}\begin{pmatrix}\theta_1\\ \dot\theta_1\\ \theta_2\\ \dot\theta_2\end{pmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0\\ -\frac{2}{J_*}k_L\frac{r_1^2 r_{p_2}^2}{r_{p_1}^2} & -\frac{c_*}{J_*} & \frac{2}{J_*}k_L\frac{r_1 r_{p_2} r_2}{r_{p_1}} & 0\\ 0 & 0 & 0 & 1\\ \frac{2}{J_2}k_L\frac{r_1 r_{p_2} r_2}{r_{p_1}} & 0 & -\frac{2}{J_2}k_L r_2^2 & -\frac{c_2}{J_2}\end{bmatrix}}_{A_c}\begin{pmatrix}\theta_1\\ \dot\theta_1\\ \theta_2\\ \dot\theta_2\end{pmatrix} + \underbrace{\begin{bmatrix}0\\ \frac{1}{J_*}k_t k_a k_c\\ 0\\ 0\end{bmatrix}}_{B_c} u + \underbrace{\begin{bmatrix} 0 & 0 & 0\\ -\frac{1}{J_*} & 0 & -\frac{2}{J_*}k_L\frac{r_1 r_{p_2}}{r_{p_1}}\\ 0 & 0 & 0\\ 0 & -\frac{1}{J_2} & \frac{2}{J_2}k_L r_2\end{bmatrix}}_{B_{w,c}}\begin{pmatrix}\Psi_*(\dot\theta_1)\\ \Psi_2(\dot\theta_2)\\ \Delta(\theta)\end{pmatrix} \tag{4.23}$$

with $\theta := \frac{r_{p_2}}{r_{p_1}} r_1\theta_1 - r_2\theta_2$, k_L is the elastic constant of the belt, J_2 and c_2 are the load inertia and viscous damping coefficient, the r_x are radii of pulleys in the problem, $\Psi_2(\dot\theta_2)$ is the nonlinear friction characteristic associated with the load, and $\Delta(\theta)$ is a nonlinearity associated with the backlash/deadzone nonlinearity in the idler pulley. In particular it is the difference between the backlash nonlinearity and the linear function that would exist if the size of the backlash was zero. The continuous-time differential equation is discretized and a feedback loop is used as before, resulting in the augmented state

$$x_k := \begin{pmatrix}\theta_1(kh)\\ \dot\theta_1(kh)\\ \theta_2(kh)\\ \dot\theta_2(kh)\\ \dot\theta_1((k-1)h)\end{pmatrix}. \tag{4.24}$$

The output for this model is a noisy measurement of both shaft angles. There are disturbance noises in the state-space equations for both x_2 and x_4. There are 3 SISO static functions in this problem; $\mathcal{S}_1$, z_1 and w_1 are defined as before, $\mathcal{S}_2$, etc are analogously defined for the load inertia quantities, and $\mathcal{S}_3 = \Delta$, with $z_3 = \theta$ and $w_3 = \mathcal{S}_3(z_3)$.

Once again we generated simulation data for the problem; Figure 4.19 shows the data used in many of the following cases. In particular we see the size of the disturbance

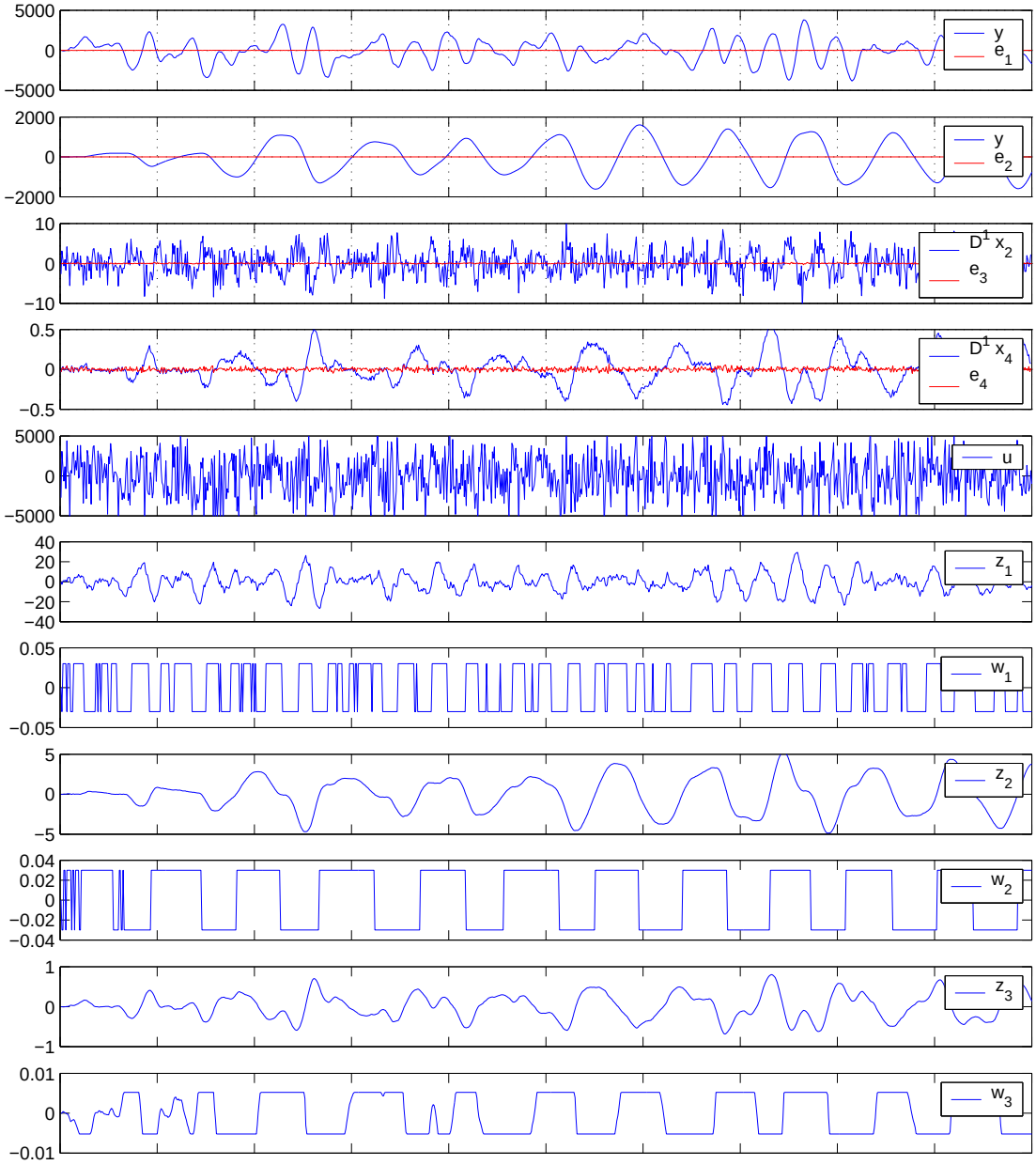


Figure 4.19: Simulation data for ECP model with load inertia.

noise level relative to the first differences of x_2 and x_4 (as before), and the inputs and outputs to the 3 static functions.

This estimation problem is quite a bit more challenging than it was for the noload system. Here there are 7 unknown signals (4 noise and 3 w) in the $\mathcal{L}_y$ set of equations, and two measured outputs, leavening a balance of 5, whereas in that problem there were three unknown signals (2 noise and 1 w) and one measured output, for a balance of 2. This is one determinant of the number of degrees of freedom associated with estimates, and how hard it might be to narrow the choices down to a good one.

Another issue now is that in using the dispersion objective there are now 3 static operators to consider. We will combine them into a single objective; this means we have to choose weights, and the issue then is to find good weighting values for that combined objective, as well as constraints for the noise signals which now number four. Our first try was to choose equal weightings for the Γ_w weights, namely $\Gamma_w(1) = \Gamma_w(2) = \Gamma_w(3) = 1$. Then guided by the expected relative size of w_i and z_i we choose $\Gamma_z(i)$ to balance the contribution of the w and z part of the dispersion; in this case we took $\Gamma_z = [0.0025\ 0.01\ 0.02]$. This led to the three scatter plot estimates shown in the color cyan in Figure 4.20. Here the estimates for $\mathcal{S}_1$ and $\mathcal{S}_3$ look reasonable, but the one for $\mathcal{S}_2$, the nonlinear friction characteristic for the load inertia, is poor, too small. Moreover, we found it was difficult to relieve some of this bias by using our standard tricks with the noise level constraint. We considered changing the relative weight of $\mathcal{S}_2$'s element in the dispersion objective, by parametrizing the dispersion weights as $\Gamma_w = [1\ \beta\ 1]$, $\Gamma_z = [0.0025\ 0.01\beta\ 0.02]$, and so we can use β to adjust the contribution of $\mathcal{S}_2$ in the objective. We find that when the size of the contribution is lowered, estimates get better, to a point. See in Figure 4.20 that for $\beta = 0.1$ the estimate is much improved. This indicates that when its weighting was high the dispersion of $\mathcal{S}_2$ was being overly penalized, resulting in the small elements, and when this downward pressure on the dispersion of $\mathcal{S}_2$ is pulled back the estimate can expand.

It also is interesting to consider the relative contributions of the outputs of the three functions on the output of the system. For instance it is apparent that if one of the functions has no effect on the output of the system, then estimates of that function will be poor, as there is no information in the experiment about it. In this example, the output w_1 of $\mathcal{S}_1$ contributes to the evolution of x_2, w_2 contributes to that of x_4, and the w_3 signal (the deadzone nonlinearity) contributes to both. Figure 4.21 shows how large each of these is relative to the first difference of x_2 and x_4, for a portion of the data record. This shows

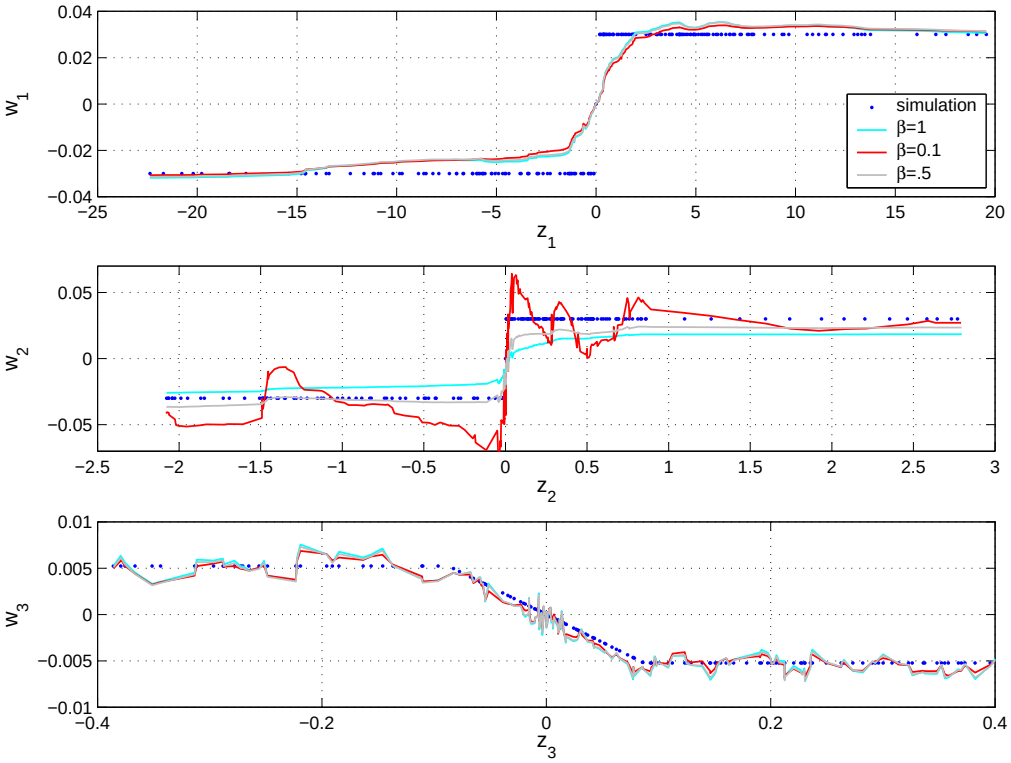


Figure 4.20: Effect of the weighting on $\mathcal{S}_2$ in the dispersion function.

that in the one place where w_2 enters, its contribution is relatively small, compared to the outputs of the other static functions. Hence its contribution to the output of the system is relatively small, and perhaps we should expect to not be able to estimate it well.

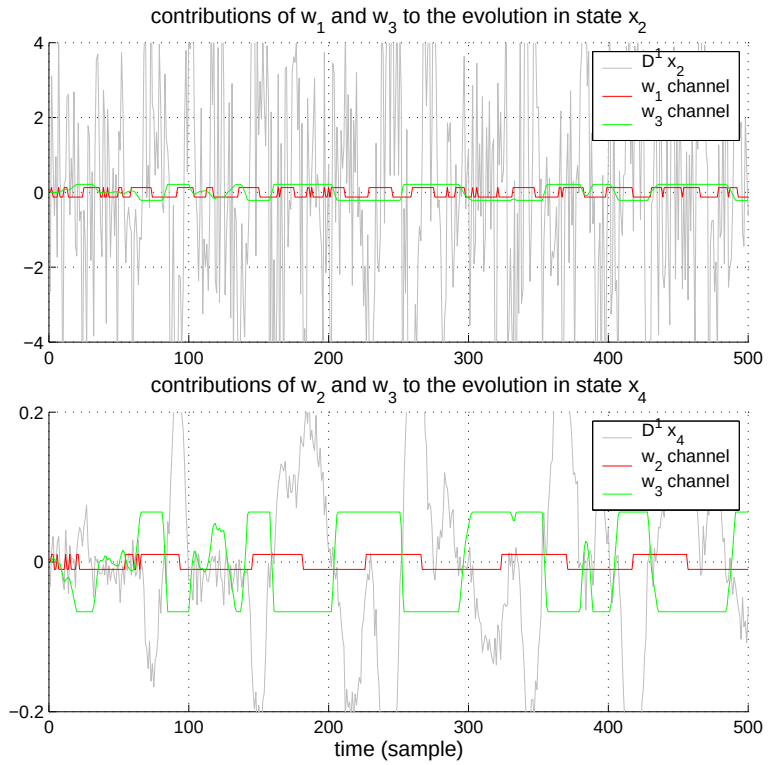


Figure 4.21: Effect of the weighting on $\mathcal{S}_2$ in the dispersion function.

Chapter 5

Error Bounds for Estimates of Static Functions

A desirable objective in problems involving identification of unknown functions is to bound the error between an estimated and actual function of the data-generating system. Often this problem is considered under the idealized circumstance that the data-generating system is contained in the identification model structure. A related question is, how does the quality of estimates depend on the model structure and the particular experiment whereby data is collected? For instance are some input signals more informative than others? We explore these issues in this chapter.

Under some rough assumptions on the data-generating system, the elements to be identified, and properties of the estimates (principally, that they are consistent with input-output data and assumed Lipschitz conditions for the unknown static functions being identified), we derive a type of bound on the mismatch between the actual and identified static nonlinearities. These bounds depend on the input used in the identification experiment, and thus they can be useful for comparing different experiments in terms of how informative they are, or ascertaining whether some degree of accuracy can be guaranteed by a particular combination of experiment, identification procedure, and apriori information. These bounds are applicable for the types of estimates produced by some of the identification techniques proposed in this thesis, among others.

5.1 Motivating Example and Main Ideas

Consider the following "sum-of-2-nonlinearities" system, as a simple example to illustrate the main ideas in this chapter for deriving and computing error bounds:

$$y(t) = \mathcal{S}_1\big(z_1(t)\big) + \mathcal{S}_2\big(z_2(t)\big), \quad t \in T. \tag{5.1}$$

The problem is to form estimates of the two unknown static SISO functions $\mathcal{S}_1$ and $\mathcal{S}_2$, or, to resolve the input-output behavior of the system into the individual contributions from these functions. To help in doing this there is experimental data available; the measured input-output data record contains $y(t)$ and $z(t)$ for times $t \in T = \{1, \ldots, L\}$. Suppose we come up with estimates $\hat{\mathcal{S}}_i(\cdot)$ over a domain $D_i \subseteq \mathbf{R}$ of $\mathcal{S}_i(\cdot)$ that is of interest, by the following general procedure.

In previous chapters we discussed ways to form estimates $\hat{w}_i$ of the signals $w_i(t) = \mathcal{S}_i(z_i(t))$ that were the outputs of the functions during the experiment. Using these estimates we define $\hat{\mathcal{S}}$ on the points where the data record "sampled" $\mathcal{S}$, as

$$\hat{\mathcal{S}}_i(z_i(t)) := \hat{w}_i(t), \quad \forall t \in T, \ i = 1, 2. \tag{5.2}$$

The experiment data itself does not contain information about $\mathcal{S}$ other than its value at these points in z; we can argue that if in the data-generating system $\mathcal{S}$ had been replaced by any other function matching $\mathcal{S}$ at z then we would have obtained the same data record. This set of points is finite since we have discrete-time systems and finite experiments.

So the second part of forming estimates is to somehow "fill in" the estimate $\hat{\mathcal{S}}_i(\cdot)$ at points of D_i that are not hit by z_i in order to form a full description of $\hat{\mathcal{S}}_i$. This choice is constrained by the assumed intersample behavior of the functions, i.e. smoothness assumptions. If the intersample behavior is arbitrary then we cannot really hope to achieve estimates with bounded error based on a finite input/output data record. However, it is often appropriate to restrict our consideration to functions which have some type of smoothness property. Generally speaking, the more smooth a function is, the more restricted its inter-sample behavior is, and the smaller the set of functions that is compatible with a given finite discrete input/output data record. In this chapter we use Lipschitz smoothness constraints in order bound intersample behavior and come up with error bounds.

Define $d_i(\cdot)$ as the difference between the estimated and true function:

$$d_i(z) := \hat{\mathcal{S}}_i(z) - \mathcal{S}_i(z). \tag{5.3}$$

It is the properties of this quantity that we are attempting to characterize or bound. Now an important feature of the estimates of $\hat{w}_i$ in preceding chapters is that they satisfy the linear system constraints, so that

$$y(t) = \hat{w}_1(t) + \hat{w}_2(t), \quad t \in T. \tag{5.4}$$

If we substitute (5.2) into this and subtract (5.1), we see the d_i satisfy

$$d_1\big(z_1(t)\big) + d_2\big(z_2(t)\big) = 0, \quad \forall t \in T. \tag{5.5}$$

In other words the difference between $\hat{\mathcal{S}}_1$ and $\mathcal{S}_1$ at the point $z_1(t)$ is the opposite of the difference between $\hat{\mathcal{S}}_2$ and $\mathcal{S}_2$ at $z_2(t)$, and this holds for all times t in the data sequence. Considering that the data record may be long and the input signals complex, this implies a set of constraints between d_1 and d_2 that is potentially large and nontrivial.

The question becomes, how large is the set of function pairs (d_1, d_2) which satisfy (5.5)? The answer depends on the inputs z_1 and z_2. To start off, the set contains the pair of identically zero functions $(0, 0)$; this just means it's possible that the estimates are equal to the actual functions. Are there any nonzero pairs in the set? If not, then for any estimates such that (5.5) holds, these estimates must be equal to the true nonlinearities.

However we quickly see that there *is* a class of nonzero pairs; consider:

$$d_1(z_1) = c \;\; \forall z_1, \quad d_2(z_2) = -c \;\; \forall z_2 \tag{5.6}$$

where c is any number. These satisfy (5.5). In other words the system equations (5.1) allow for estimates which are equal and opposite translations of the true nonlinearities by an arbitrary amount. Nor will a smoothness constraint rule out a constant translation, since a constant function is quite benign and adding a constant will not change most smoothness measures. Given no other information about the true nonlinearities there is really no way to rule out constant translations. This is an identifiability issue.

We could require that additional information about the true functions be provided so that we can distinguish estimates which are constant translations. For instance maybe the value of $\mathcal{S}_1(z^o)$ is known, and so we require that $\hat{\mathcal{S}}_1(z^o) = \mathcal{S}_1(z^o)$. In this case the constant translation is resolved and the question becomes, how far away is the mismatch from zero, and a specific measure of that, for $\mathcal{S}_1$, is

$$\max_{z \in D_1} |d_1(z)| \tag{5.7}$$

Alternatively, we could take the attitude that a constant translation of $\mathcal{S}$ is just as good an estimate as $\mathcal{S}$ itself. In this case a relevant question to ask is, how far away can d_1 and d_2 be from being constant? A specific measure of this quality, for $\mathcal{S}_1$, might be something like

$$\max_{z_a \in D_1} d_1(z_a) - \min_{z_b \in D_1} d_1(z_b) = \max_{z_a, z_b \in D_1} |d_1(z_a) - d_1(z_b)|. \tag{5.8}$$

The approach in this chapter encompasses both of these approaches.

To keep notation simpler, for the time being let's restrict our attention to estimates and bounds for the first function, $\mathcal{S}_1$. (Since the problem is symmetric in the two functions it's a simple matter to modify the development for the second function.) The subscript "1" on some of the named quantities that follow reflects this choice.

Now assume that $\mathcal{S}_1$ and $\mathcal{S}_2$ are Lipschitz continuous, i.e. there exist γ_1, γ_2 such that

$$\mathcal{S}_1 \in L_{\gamma_1}, \quad \mathcal{S}_2 \in L_{\gamma_2}. \tag{5.9}$$

Assume as well that the estimates $\hat{\mathcal{S}}_i$ also satisfy this Lipschitz condition. When using the methods described in chapters 2, 3, and 4 to estimate $\hat{w}$ we can ensure explicitly that the partial estimates (5.2) satisfy the Lipschitz condition (3.125) for pairs of points in z_i by using Lipschitz constraints as described in Section 3.3.4 in the estimation problem formulation. With these assumptions, for any two times t, s in the data record we have

$$\begin{aligned}
\Big|d_1(z_1(t)) - d_1(z_1(s))\Big| &= \Big|\{\hat{\mathcal{S}}_1(z_1(t)) - \mathcal{S}_1(z_1(t))\} - \{\hat{\mathcal{S}}_1(z_1(s)) - \mathcal{S}_1(z_1(s))\}\Big| \\
&= \Big|\{\hat{\mathcal{S}}_1(z_1(t)) - \hat{\mathcal{S}}_1(z_1(s))\} - \{\mathcal{S}_1(z_1(t)) - \mathcal{S}_1(z_1(s))\}\Big| \\
&\leq 2\gamma_1 |z_1(t) - z_1(s)|.
\end{aligned} \tag{5.10}$$

We will refer to this rather straightforward bound on the difference in the error at two points as the *direct Lipschitz bound*. It is a direct consequence of $\mathcal{S}_1, \hat{\mathcal{S}}_1 \in L_{\gamma_1}$, and depends on γ_1 only. In general we can use our knowledge of the system equations and z to find a less conservative bound. Since $d_1(z_1(t)) = -d_2(z_2(t)) \;\; \forall t \in T$, the difference can also expressed in terms of $\mathcal{S}_2$ and $\hat{\mathcal{S}}_2$ and bounded as

$$\begin{aligned}
\Big|d_1(z_1(t)) - d_1(z_1(s))\Big| &= \Big|d_2(z_2(t)) - d_2(z_2(s))\Big| \\
&= \Big|\{\hat{\mathcal{S}}_2(z_2(t)) - \mathcal{S}_2(z_2(t))\} - \{\hat{\mathcal{S}}_2(z_2(s)) - \mathcal{S}_2(z_2(s))\}\Big| \\
&= \Big|\{\hat{\mathcal{S}}_2(z_2(t)) - \hat{\mathcal{S}}_2(z_2(s))\} - \{\mathcal{S}_2(z_2(t)) - \mathcal{S}_2(z_2(s))\}\Big| \\
&\leq 2\gamma_2 |z_2(t) - z_2(s)|.
\end{aligned} \tag{5.11}$$

Choosing the the stronger of(5.10) and (5.11) we have

$$\Big|d_1(z_1(t)) - d_1(z_1(s))\Big| \leq \min\Big\{2\gamma_1|z_1(t) - z_1(s)|\ ,\ 2\gamma_2|z_2(t) - z_2(s)|\Big\}\ \ \forall t, s \in T. \quad (5.12)$$

What can happen here is that in the data there may be a pairs of times (t, s) such that $z_1(t)$ and $z_1(s)$ are relatively far apart, so that the direct Lipschitz bound is quite large, but it so happens that $z_2(t)$ and $z_2(s)$ are very close. In this case $2\gamma_2|z_2(t) - z_2(s)|$ can provide a small bound on the difference between values of d_1 for points spaced far apart, relative to the direct bound based on those points. This is the main thing we are trying to take advantage of. It results directly from (5.5), which reflects the constraint that the estimates satisfy the input/output data. To Look at it in another way, z_2 might not be very different at time t than at time s, while the difference for z_1 at those two times is bigger. Then, in view of the Lipschitz bounds, most of the change in y must be explained by $\mathcal{S}_1$, and we get strong information about the relationship between $\mathcal{S}_1(z_1(t))$ and $\mathcal{S}_1(z_1(s))$.

Next, the bound given by (5.12) can generally be improved by applying the triangle inequality. First define some notation for the RHS of (5.12),

$$k_1(s, t) := \min\Big\{2\gamma_1|z_1(t) - z_1(s)|\ ,\ 2\gamma_2|z_2(t) - z_2(s)|\Big\}.$$

This bound on $|d_1(z_1(t)) - d_1(z_1(s))|$ is defined for all $t, s \in T$. Now fix $t, s \in T$. For any subset $\{r_1, r_2, \ldots, r_n\}$ of T,

$$\begin{aligned}\Big|d_1(z_1(t)) - d_1(z_1(s))\Big| &\leq \Big|d_1(z_1(t)) - d_1(z_1(r_1))\Big| \\ &\quad + \sum_{i=1}^{n-1}\Big|d_1(z_1(r_i)) - d_1(z_1(r_{i+1}))\Big| + \Big|d_1(z_1(r_n)) - d_1(z_1(s))\Big|, \quad (5.13)\end{aligned}$$

and so

$$\Big|d_1(z_1(t)) - d_1(z_1(s))\Big| \leq k_1(t, r_1) + \sum_{i=1}^{n-1} k_1(r_i, r_{i+1}) + k_1(r_n, s) \quad (5.14)$$

We should search for the best bound on $|d_1(z_1(t)) - d_1(z_1(s))|$ by searching over all "paths" $\{r_1, r_2, \ldots, r_n\}$ through $k_1(s, t)$ (which itself is a function of the signals z_i and the Lipschitz constants).

This is a shortest-path graph problem; the graph in question has nodes at the time points $t \in T$, and a weighting between nodes s and t equal to $k_1(s, t)$. To compute the best bound on $|d_1(z_1(t)) - d_1(z_1(s))|$ we look for the shortest path between t and s through this graph. Let $l_1(s, t)$ denote the length of this path. There is an efficient algorithm to

simultaneously compute the shortest path between each pair of nodes in a graph, which is commonly called the all-pairs shortest path (APSP) problem. Specifically, the Floyd-Warshall algorithm computes $l_1(\cdot,\cdot) : T \times T \to \mathbf{R}$ in $O(N^3)$ time, where N is the number of nodes (Papadimitriou and Steiglitz 1982). In our case $N = L$, the length of the data record.

At this point we have computed a function $l_1(\cdot,\cdot) : T \times T \to \mathbf{R}$, a bound on the difference of the error in estimating $\mathcal{S}_1$, defined for pairs of points in the data record for the z_1 signal. Now let us switch notation slightly, changing the domain of $k_1(\cdot,\cdot)$ and $l_1(\cdot,\cdot)$ from pairs of times in the data record to pairs of points in the z_1 data record, using the obvious one-to-one correspondence between the two. Thus we have

$$|d_1(z_a) - d_1(z_b)| \leq l_1(z_a, z_b) \quad \forall z_a, z_b \in Z_1, \tag{5.15}$$

where $Z_1 := \{z_1(t) : t \in T\} \subseteq D_1$, the z_1 data record. Using $l_1(\cdot,\cdot)$ it is straightforward to extend the bound to arbitrary pairs of points in D_1 using the Lipschitz constraint on $\mathcal{S}_1$, the assumption that the estimation method satisfies this constraint when filling in intersample points, and the triangle inequality. One bound is provided, as usual, by the direct Lipschitz bound. Next, apply the bounds provided by $l_1(\cdot,\cdot)$ by "connecting" z_a and z_b with points in Z_1. If we connect z_a to $z_{11} \in Z_1$ and z_b to $z_{12} \in Z_1$ then the triangle inequality implies

$$|d_1(z_a) - d_1(z_b)| \leq 2\gamma_1|z_a - z_{11}| + l_1(z_{11}, z_{12}) + 2\gamma_1|z_b - z_{12}|. \tag{5.16}$$

Due to the nature of the graph search step used in constructing l_1 we need only consider connecting z_a and z_b to the nearest upper and lower neighboring points from Z_1 (for other points the bound of (5.16) is guaranteed to be larger). Let $z_{a,L}$ be the greatest element of Z_1 which is less than or equal to z_a and $z_{a,U}$ the least element that is greater than or equal to it, and similarly $z_{b,L}$ and $z_{b,U}$ are the nearest elements of Z_1 that surround z_b. Putting these possibilities together, the bound for arbitrary $(z_a, z_b) \in D_1 \times D_1$ is

$$\begin{aligned}
|d_1(z_a) - d_1(z_b)| \leq \min\{&2\gamma_1|z_a - z_b|, \\
&2\gamma_1|z_a - z_{a,L}| + l_1(z_{a,L}, z_{b,L}) + 2\gamma_1|z_b - z_{b,L}|, \\
&2\gamma_1|z_a - z_{a,L}| + l_1(z_{a,L}, z_{b,U}) + 2\gamma_1|z_b - z_{b,U}|, \\
&2\gamma_1|z_a - z_{a,U}| + l_1(z_{a,U}, z_{b,L}) + 2\gamma_1|z_b - z_{b,L}|, \\
&2\gamma_1|z_a - z_{a,U}| + l_1(z_{a,U}, z_{b,U}) + 2\gamma_1|z_b - z_{b,U}|\}.
\end{aligned} \tag{5.17}$$

Let's denote this quantity, the extended bound, as $m_1(z_a, z_b) : D_1 \times D_1 \to \mathbf{R}$. (An alternative way to extend bounds beyond pairs in Z_1 would have been to add additional points in D_1 to k_1 using the direct Lipschitz bound to connect the additional points to existing points in k_1's domain, and then use the FW algorithm to produce an m_1 bound containing those points. The result is the same.)

With this quantity in hand we can get back to the task at hand as described on page 145. The quantity (5.8), measuring the spread or non-constantness of the error, is bounded by

$$\max_{z_a, z_b \in D_1} |d_1(z_a) - d_1(z_b)| \leq \max_{z_a, z_b \in D_1} m_1(z_a, z_b). \tag{5.18}$$

For instance if $m_1(z_a, z_b) = 0 \;\; \forall z_a, z_b$, then the error must be constant. The bound is also applicable in the case additional information is known, such as the value of $\mathcal{S}_1(z^o)$, and we're interested in the max error as in (5.7). In that case $d_1(z^o) = 0$, and so

$$\max_{z_1 \in D_1} |d_1(z_1)| \leq \max_{z_1 \in D_1} m_1(z^o, z_1), \tag{5.19}$$

and now if $m_1 = 0$ everywhere then the estimate must equal the true function.

5.2 Some Features of Bounds and Estimates

We now move on to discuss some properties of these bounds, in particular how the bounds achieved depend on the input signal. First consider a data record in which z_2 is held at a single fixed value z_2^o throughout, and z_1 varies. If $\mathcal{S}_2(z_2^o)$ is known then in view of (5.1) we in effect have a perfect measurement of $\mathcal{S}_1(z)$ for all $z \in Z_1$, that is wherever that function is sampled. If $\mathcal{S}_2(z_2^o)$ is not known then we still essentially know $\mathcal{S}_1(z)$ at these points, modulo a constant translation. The data record explores $\mathcal{S}_1$ independently of $\mathcal{S}_2$, since z_1 varies independently of z_2. On the other hand $\mathcal{S}_2$ is in some sense identified very poorly. We only have information about it at a single point in its domain, z_2^o. To the extent that z_1 varies at successive points where z_2 doesn't vary much you get good information about $\mathcal{S}_1$ but not so much about $\mathcal{S}_2$, and vice versa. Note that for the static example we have been working with, this "independent variation" can apply to non-consecutive time steps, for instance if the variance of the set $\{z_1(t) : t \in U\}$ is much larger than the variance of the set $\{z_2(t) : t \in U\}$, for an arbitrary subset $U \subseteq T$. The importance of this effect appears in (5.12), where $|z_2(t) - z_2(s)|$ is used to bound $|d_1(z_1(t)) - d_1(z_1(s))|$; to the extent that z_1 varies while z_2 is constant, this statement is more powerful.

To some extent there is a tradeoff. Some inputs may do a good job providing information about $\mathcal{S}_1$ but not $\mathcal{S}_2$, some may be more useful for identifying $\mathcal{S}_2$ than for identifying $\mathcal{S}_1$. Next consider any data record in which $z_2 = z_1$. In this case we observe

$$y(t) = \mathcal{S}_1(z_1(t)) + \mathcal{S}_2(z_1(t)) \tag{5.20}$$

$$= \underbrace{\mathcal{S}_1(z_1(t)) + v(z_1(t))}_{=:\tilde{\mathcal{S}}_1(z_1(t))} + \underbrace{\mathcal{S}_2(z_1(t)) - v(z_1(t))}_{=:\tilde{\mathcal{S}}_2(z_1(t))}, \tag{5.21}$$

where here $v(\cdot)$ is an arbitrary function. If $\tilde{\mathcal{S}}_1(\cdot)$ and $\tilde{\mathcal{S}}_2(\cdot)$ meet the Lipschitz constraints and any known information, then it is impossible to distinguish, on the basis of this data record, that the estimates $\tilde{\mathcal{S}}_1$ and $\tilde{\mathcal{S}}_2$ are any less valid than $\mathcal{S}_1$ and $\mathcal{S}_2$ themselves, or constant translations of them, since they satisfy the data record and assumptions. Since v may vary widely and still satisfy these constraints, we cannot hope to arrive at meaningful estimates. The situation is sort of worst-case in terms of the issue described in the previous paragraph. Neither function's input varies while the other stays fixed. So some data records may be especially poor in that they identify neither function well. In these cases the experiment lacks the power to resolve the output into the individual contributions from the two nonlinearities. For this problem if the functions' inputs are not truly independent and are in fact always equal, then the model should be using only one function instead of two. This is an identifiability issue. However if the two functions' inputs are actually independent in that they are not necessarily identical, then we need to have distinct inputs to resolve that there are two separate nonlinearities involved.

Another characteristic of the input data that is plainly important is that of how densely it covers the domain of interest. Typically we are concerned with identifying a function over some infinite domain, for instance when D_1 is an interval in $\mathbf{R}$. The input-output data can only give information about the function at a finite subset of that domain and for the remaining portions we essentially must make a guess. The guess is constrained by assumed smoothness criteria, and in the present scenario this is the Lipschitz constraint on the function. This assumption limits how far off the guess might be, and this is what we use to supply error bounds involving these regions. As noted, this is a very simple and conservative bound. The material of this chapter is an effort to provide less conservative bounds, but in regions where input data is sparse we must resort to cruder bounds. Therefore inputs with large regions lacking points in the input data record will tend to produce worse bounds.

To summarize, there are at least two characteristics of inputs that affect the bounds on estimate quality of e.g. $\mathcal{S}_1$. One is how well the input data record z_1 covers the domain D_1 where we wish to identify the function. Another is the extent to which one input varies independently of the other.

Equation (5.18) or (5.19) boils the information in m_1 down into one number that bounds the variation of the error over the domain D_1 or the maximum absolute error over D_1, respectively, based on a given input data record. This is a valid bound, but m_1 contains more information. Recall $m_1(z_a, z_b)$ is a bound on the difference between the estimation errors at z_a at z_b, and is therefore some kind of measure on how much the error can differ from being a constant. Now, because m_1 contains a bound for each pair of points in D_1 it can alert us to problem regions and give clues as to the weaknesses of a particular input. For instance the data record may cover some regions of D_1 relatively sparsely, and as we pointed out above this will generally lead to less impressive bounds for pairs of points involving these regions, and therefore a larger entire-domain bound. Another situation is that the data may contain a small number of "outlier" points, which are far away from and essentially independent of the other data points. These are contrary to the holding-one-fixed portion of the holding-one-fixed-while-the-other-varies notion, and can make it harder to identify $\mathcal{S}$ at these points. Now, if we are willing to exclude these few points or regions from D_1, then entire-domain bounds such as (5.18) can become much tighter over the slightly smaller domain. Thus, it can be useful to look at the complete picture painted by the two-dimensional $m_1(\cdot, \cdot)$, and can provide guidance to maybe exclude some points, redesign the experiment, or look at other possibilities to obtain a tighter bound on the estimation error (and presumably also, better estimates).

Now we consider two examples that illustrate some of these issues. In both the model being considered is that of (5.1), the sum of two unknown functions, a static system with no measurement noise.

In "Example 1" we consider two data records, denoted A and B, and compare the bounds that result. In record A the two inputs z_1^A, z_2^A are realizations of IID noise, uniformly distributed on $[-1, 1]$, that is

$$z_i^A \sim U[-1, 1], \tag{5.22}$$

with z_1^A and z_2^A independent. In record B the input signals are similar except that the

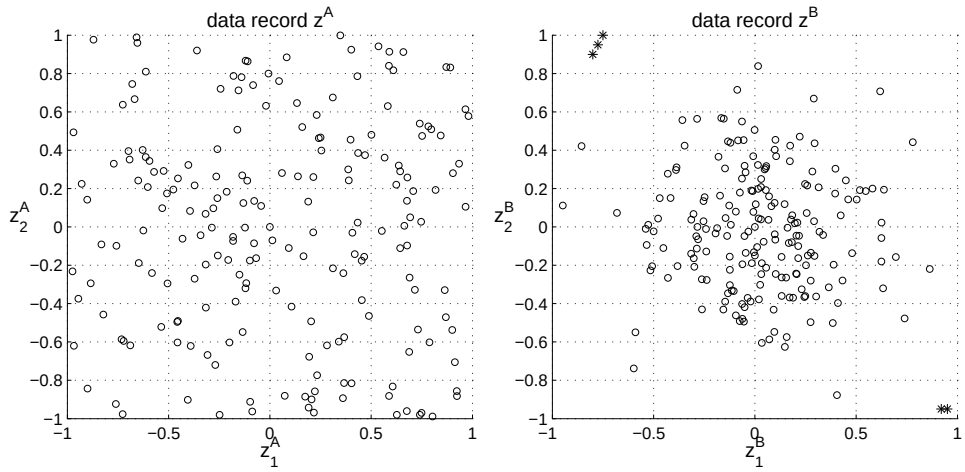


Figure 5.1: Example 1: Input data records z^A and z^B.

distribution is Gaussian, specifically

$$z_i^B \sim \mathcal{N}(0, 0.1), \tag{5.23}$$

with the variance chosen so most data falls within $[-1, 1]$. In addition, 5 non-random "outlier" points are included to demonstrate their effect. Figure 5.1 displays the first 200 points in these data records; e.g. the left panel is a plot of z_2^A vs z_1^A. In the right panel the outlier points are indicated with an asterisk. These points are relatively far away from both other z_1^B and z_2^B points. Note that the normal distribution has higher density near zero, and lower density away from zero.

We look at bounds on the error of $\mathcal{S}_1$; results for $\mathcal{S}_2$ look similar due to the qualitatively symmetric nature of the input data record. The domain of interest is $D_1 = [-1, 1]$. For both input records, and using Lipschitz constants $\gamma_1 = \gamma_2 = 1$, we compute the k_1 and l_1 bounds on the data record, and the m_1 bound over D_1. Figure 5.2 and Figure 5.3 are plots of $m_1(\cdot, \cdot)$ using the first 200 points of input records A and B, respectively. In these plots of 3-dimensional data, the x-y position locates the pair of points whose error spread is being bounded, and the color at that point corresponds to the bound. Since $m_1(z_a, z_b)$ must equal $m_1(z_b, z_a)$, the plots are symmetric. Note the difference in the character of two plots. The bound from z^A has more or less uniform features over the whole interval D_1, while that from z^B has large regions where the pairwise bound is small, and other regions where the pairwise bound involving points in those regions is much larger. The overall bound, as in (5.18), is 0.17 using z^A, and 0.42 using z^B. However when the region under

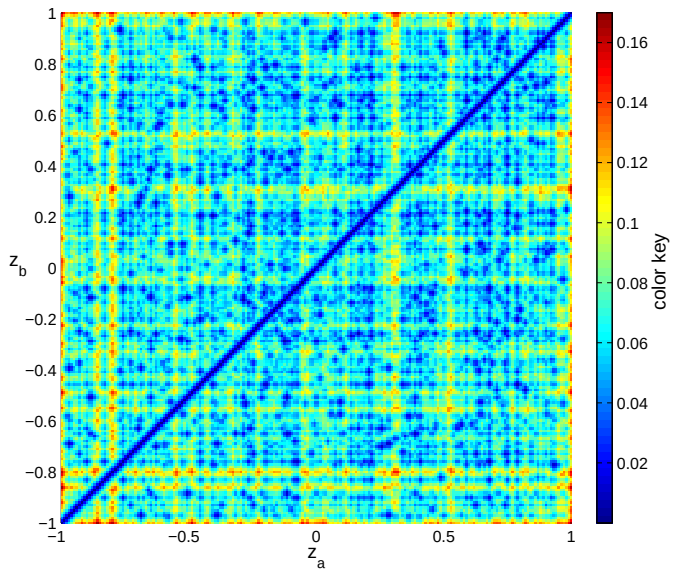


Figure 5.2: Example 1: $m_1(z_a, z_b)$ bound for z^A, uniformly distributed.

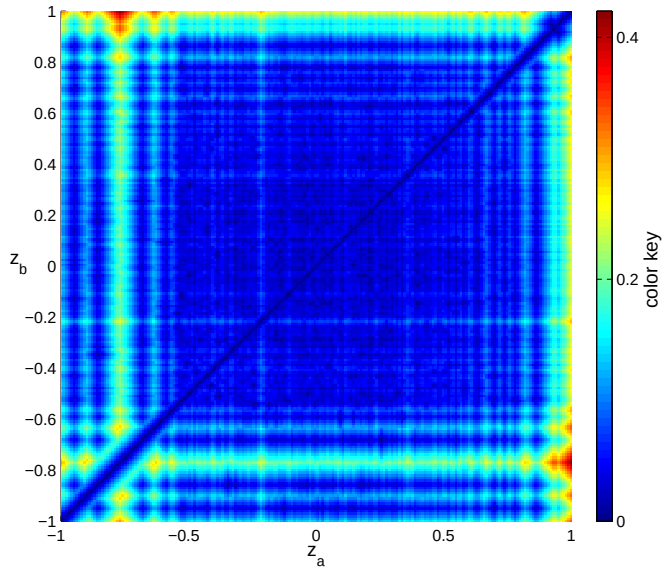


Figure 5.3: Example 1: $m_1(z_a, z_b)$ bound for z^B, normally distributed.

consideration is reduced to $D_1^* = [-0.5, 0.5]$, the bound using z^B is much lower at 0.10, while the bound using z^A, at 0.12, is much less changed, and indeed now higher than the z^B bound. Eliminating the outer regions improves the overall bound for using z^B mainly because the density of z^B is lower in these regions and so the bounds are not as good.

This shows that it can be useful to look at the picture painted by the entire collection of pairwise bounds as represented by l_1 or m_1, in addition to the overall bound (5.18). Having said this, for the remainder of the examples in this chapter, we will plot representative cross-sections of this function simply because 2-dimensional data is easier to visualize than 3-dimensional. If we hold one argument of m_1 fixed at say $\bar{z}_b$, then as z_a varies $m_1(z_a, \bar{z}^b)$ is the bound on the error difference between $\bar{z}_b$ and any point in D_1. The cross-sections we choose will be representative or indicative of the behavior we are trying to illustrate. In the remainder of the chapter we will use $D_1 = [-1, 1]$, and choose inputs that more or less fill this set with a uniform density, so that different possible cross-sections have relatively similar characteristics. If $\bar{z}_b$ is a point where $\mathcal{S}_1(\bar{z}_b)$ is known, so that the estimate error is zero there, then $m_1(z_a, \bar{z}_b)$ is a bound on the error on the domain, i.e. $|\mathcal{S}_1(z_a) - \hat{\mathcal{S}}_1(z_a)| \leq m_1(z_a, \bar{z}_b)$.

Let's look at the $\bar{z}_b = 0$ cross-sections of the bounds in the preceding example. Refer to Figure 5.4. In the top two panels the solid line is the m_1 bound, the circles are the points $l_1(z_1(t), 0)$, and the '+' symbols indicate the points $k_1(z_1(t), 0)$. The latter two are the pairwise bounds between $z_1 = 0$ and the other $L - 1$ points in the z_1 data record (we arranged so that zero is actually in that record). First note that the l_1 bound is usually much smaller than the k_1 bound; the shortest-path graph search is an essential step. Regarding the m_1 bound we see once again in these cross-sections the behavior described above, where it quickly gets larger in regions where the data record is relatively sparse. This is especially noticeable in the outer regions of $[-1, 1]$ in the z^B bounds, but in these plots it is now also noticeable in the z^A bounds, for instance near $z_a = -0.8$, and at more interior regions in the z^B bounds, for instance near $z_a = -0.2$.

The m_1 bound tends to be good wherever there is data, and in-between it grows at the rate dictated by direct application of the Lipschitz assumption, as in (5.10). The exception to this general rule is around the 5 "outlier" points that were added to the z^B data record, indicated in the second panel of Figure 5.4 using asterisks instead of circles. Unlike other points in the data record, these are not bringing the bound down any more than the direct Lipschitz constraint. This is because the input at those time points is far

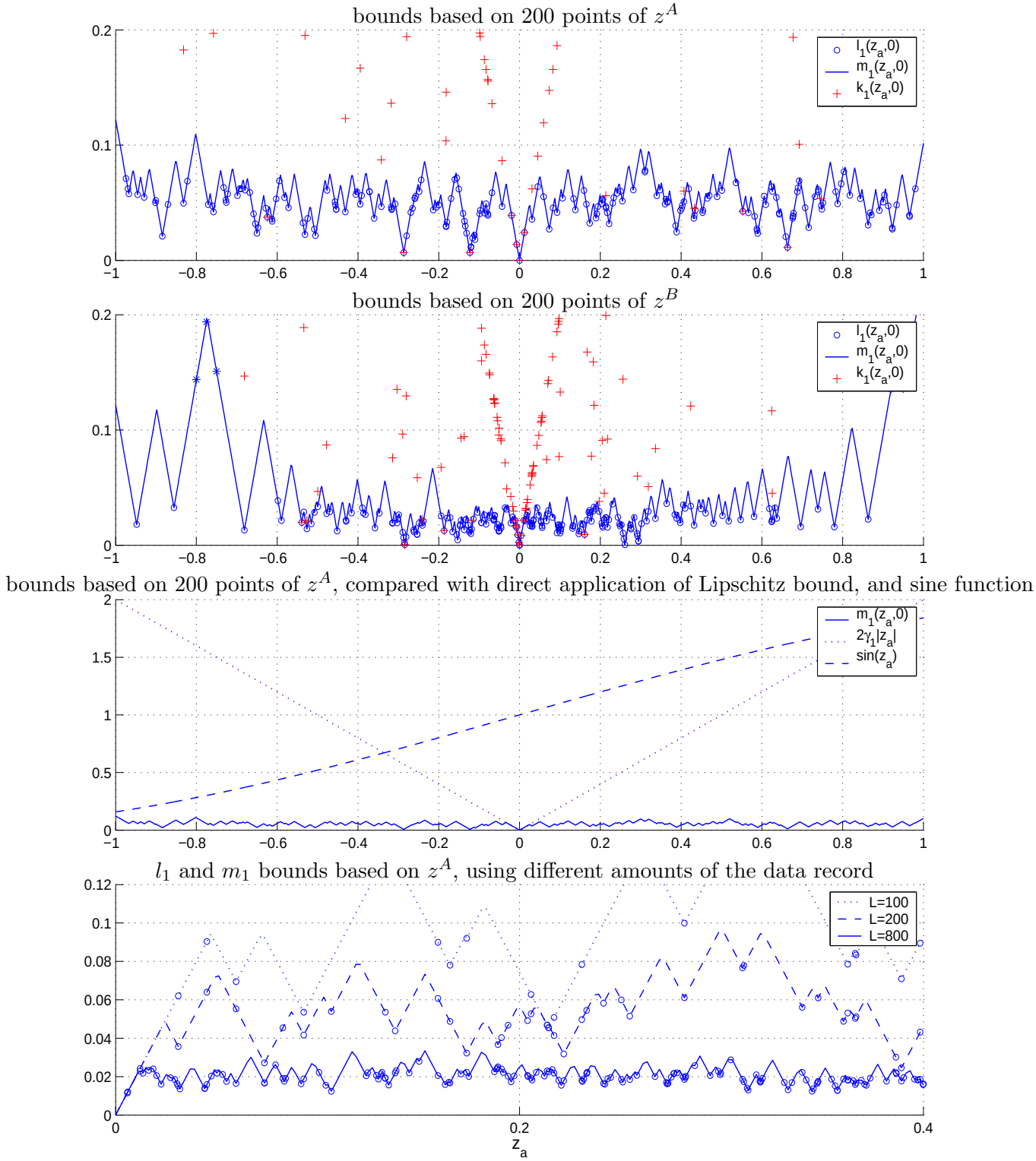


Figure 5.4: Example 1: Cross-sections of pairwise bound functions using z^A and z^B.

away from any other input points, both in z_1 and z_2, and so the key expression (5.12) isn't helping to bound the d_1 difference involving those points. Data records with good density over the domain of interest is necessary for good bounds, but not sufficient.

To place the level of the bound in perspective, in the third panel of Figure 5.4 we show the $m_1(z_a, 0)$ bound for the z^A input, along with the direct Lipschitz bound of (5.10) and a representative function that satisfies the Lipschitz assumption, namely the sine function (which is in L_{γ_1} with $\gamma_1 = 1$). In this context the bound appears relatively small. Also, we see that if we had made a Lipschitz assumption that was overly conservative, say for instance $\gamma_1, \gamma_2 = 2$ whereas the functions being estimated have Lipschitz constant 1, the resulting bound can still be useful, that is, small relative to the size of functions being estimated. For, following the derivation (5.10) through (5.17), if we compute bounds using two different sets of Lipschitz constants that differ by a common factor, then the bounds will differ by that factor as well. In Figure 5.4 it's apparent the bound could be scaled up significantly before it becomes large relative to the example nonlinearity.

Finally in the bottom panel of Figure 5.4 we show the bound computed using different amounts of the data record z^A, specifically, the first 100 time points, the first 200, and the first 800. The bound becomes better as more data is used. This is attributed to two effects. First, with more data present in the fixed region $D_1 = [-1, 1]$, the spacing necessarily becomes more dense, and so the bound in regions between data record points gets better. But notice the bound also improves for those points common to the three data records as the length increases. This is because the extra points in this randomly generated input data allow more chance for z_1 to vary independently of z_2.

In "Example 2" we look at a parametrized series of data records, observing how the parameter affects the bounds and then comparing these bounds to actual estimates of the unknown signals w_i, using the methods of the previous chapters, for two different estimation formulations. We use four sets of inputs, each of length 200, defined as

$$z_2^k(t) = -1 + 0.01t \tag{5.24a}$$

$$z_1^k(t) = \left(\!\!\left(z_2^k(t) + \alpha_k X(t) \right)\!\!\right)_{[-1,1]}, \tag{5.24b}$$

with $\alpha_1 = 0$, $\alpha_2 = 0.05$, $\alpha_3 = 0.1$ and $\alpha_4 = 0.5$. Here $X \sim U[-1, 1]$ is a uniformly distributed random variable (the same realization is used in all four), and $((\cdot))_{[-1,1]} : \mathbf{R} \to [-1, 1]$ is a modulo operator which maps $\mathbf{R}$ into the the interval $[-1, 1]$. These four input records are plotted in Figure 5.5. Each of these 8 signals varies between -1 and 1. The z_2

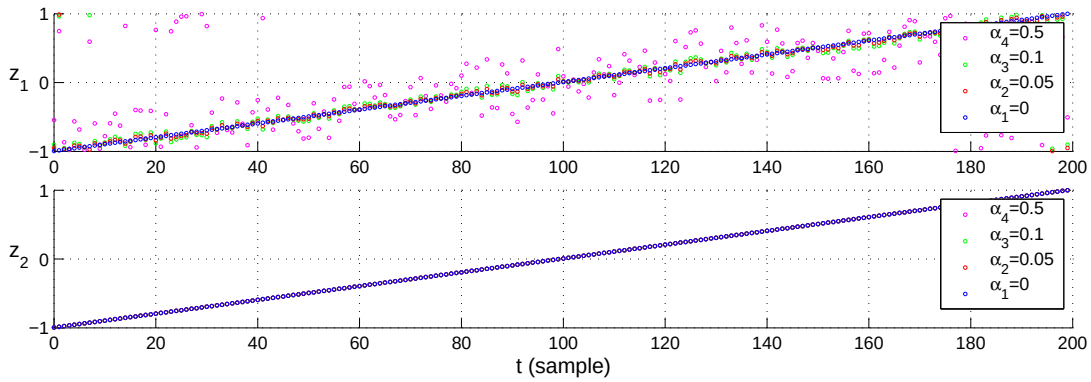


Figure 5.5: Example 2: The four inputs.

signal is the same for all four, a steady ramp of evenly spaced points going from -1 to 1. The z_1 signal is equal to z_2 plus an added fluctuation about it which is proportional to α_k, zero for the first data record and increasing for the other three. Thus in the first record there is no independence between the two input signals, while for the others the level of independence increases, in that for a given range in z_2 the z_1 signal has a larger spread, as α_k increases.

Using $\gamma_1 = \gamma_2 = 1$ the m_1-bound is computed as usual for $\mathcal{S}_1$. The result, for each of the 4 inputs, is plotted in the top panel of Figure 5.6. As expected, as the amount of "jitter" increases the bound becomes stronger. For the first data record where $z_1 = z_2$ the bound is especially poor.

Next we form estimates using these inputs. Data records are obtained by simulation. The two nonlinearities to be identified are shown in Figure 5.7. Note that both functions are in L_γ with $\gamma = 1$. The output part of the data record is simulated using these functions, according to the system (5.1). Using this estimates $\hat{w}_i$ of the signals w_i are then formed using the ideas from Chapter 3, in two formulations. To relate the current example to the setup and notation there, note that this problem contains no noise (no e signal), $u_1 = z_1$ and $u_2 = z_2$, and the linear system is defined by $y = w_1 + w_1$.

In the first estimation formulation the 2-norm of $\hat{w}$ is minimized subject to the Lipschitz constraint and the additional constraint $\mathcal{S}_1(0) = 0.8$ (which resolves the constant-translations issue and makes $m(z, 0)$ a bound on the absolute estimation error, that is

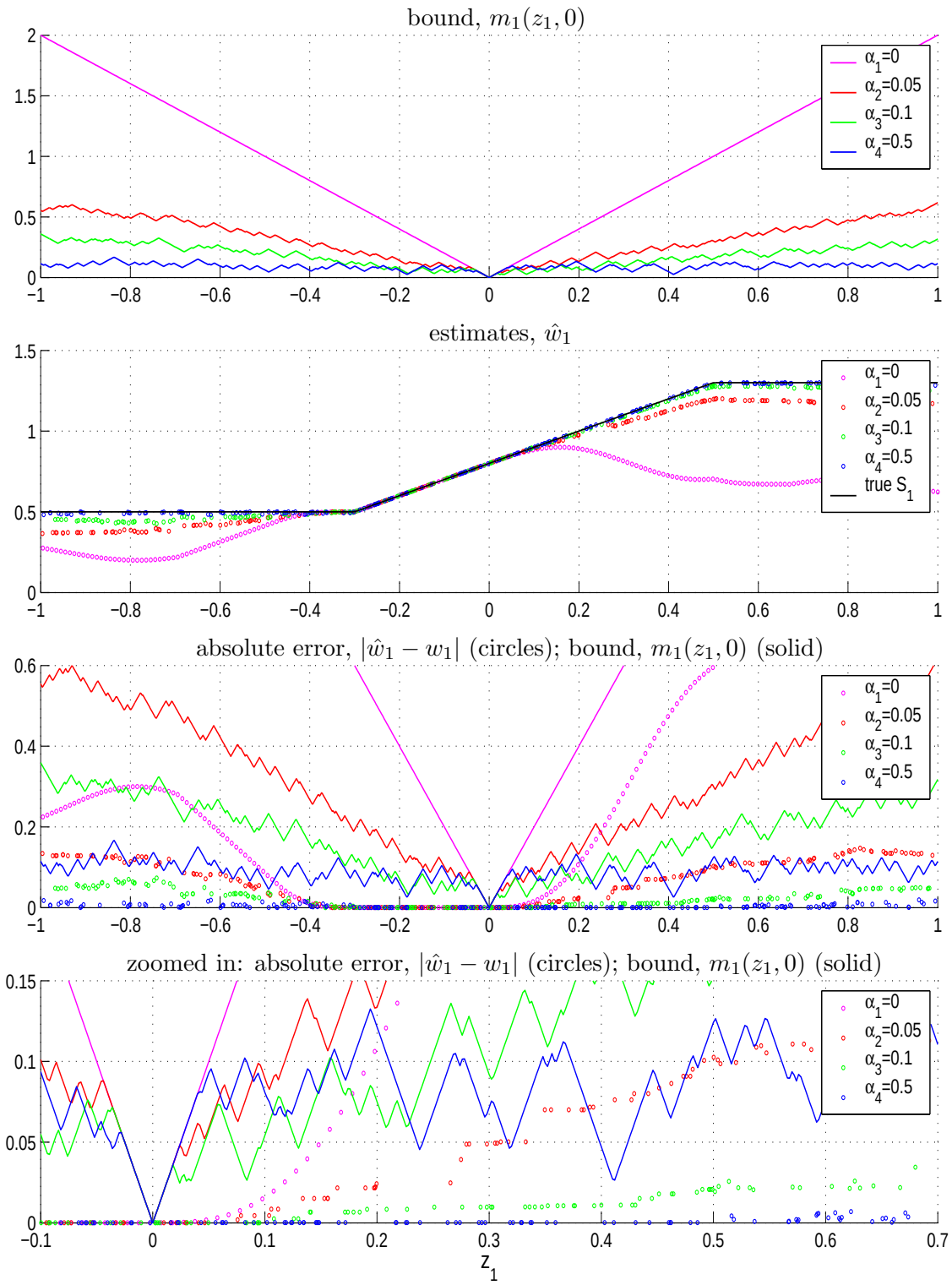


Figure 5.6: Example 2: Bounds and formulation A estimates of $\mathcal{S}_1$, for the four inputs.

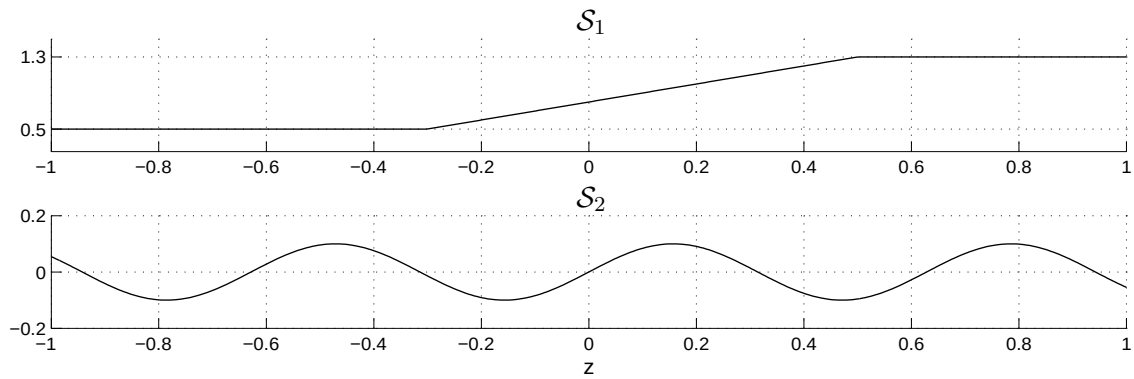


Figure 5.7: Example 2: $\mathcal{S}_1$ and $\mathcal{S}_2$.

$|w_1(z) - \hat{w}_1(z)| \leq m(z, 0)$):

$$\hat{w} \;=\; \begin{array}{ll} \underset{w}{\text{argmin}} & \|w\|_2^2 = \|w_1\|_2^2 + \|w_2\|_2^2 \\ \text{subject to} & y(t) = w_1(t) + w_2(t) \;\; \forall t \in T \\ & \mathcal{S}_1(0) = 0.8 \\ & |w_i(t) - w_i(s)| \leq |z_i(t) - z_i(s)| \;\; \forall t, s \in T. \end{array} \tag{A}$$

Scatter plots of the estimates $\hat{w}_1$ versus z_1, using each of the four input records, are shown in the second panel of Figure 5.6, along with the actual $\mathcal{S}_1$. The absolute value of the estimation errors are plotted in the third and fourth panels, along with the computed bounds (the fourth panel is the same data as the third, zoomed in to show one portion of the plot in greater detail). First off, in the third and fourth panels we see the computed m_1 indeed bounds the estimation errors, which we expect due to the Lipschitz constraint in the estimation problem formulation. In fact in this case the bounds are quite conservative. But we also see that as the input jitter α increases, the trend of the size of the estimation error is the same as that of the bounds, in that it gets uniformly smaller across $[-1, 1]$. When using the first data record in which z_1 and z_2 are identical the estimates are especially poor, and when using the fourth, the estimates look quite accurate.

Similar bounds and estimation results are obtained for the second function, $\mathcal{S}_2$. The m_2 bound and estimates $\hat{w}_2$ are shown in Figure 5.8, analogously to Figure 5.6. The overall picture for $\mathcal{S}_2$ looks look similar to the that of $\mathcal{S}_1$. The quality of the bounds and estimates for the four data records follows the same trend as for the first nonlinearity. This is not unexpected; note that the inputs z_1 and z_2 are essentially symmetric. We would see this if we plotted again the data of Figure 5.5, but reordering the time points in order of

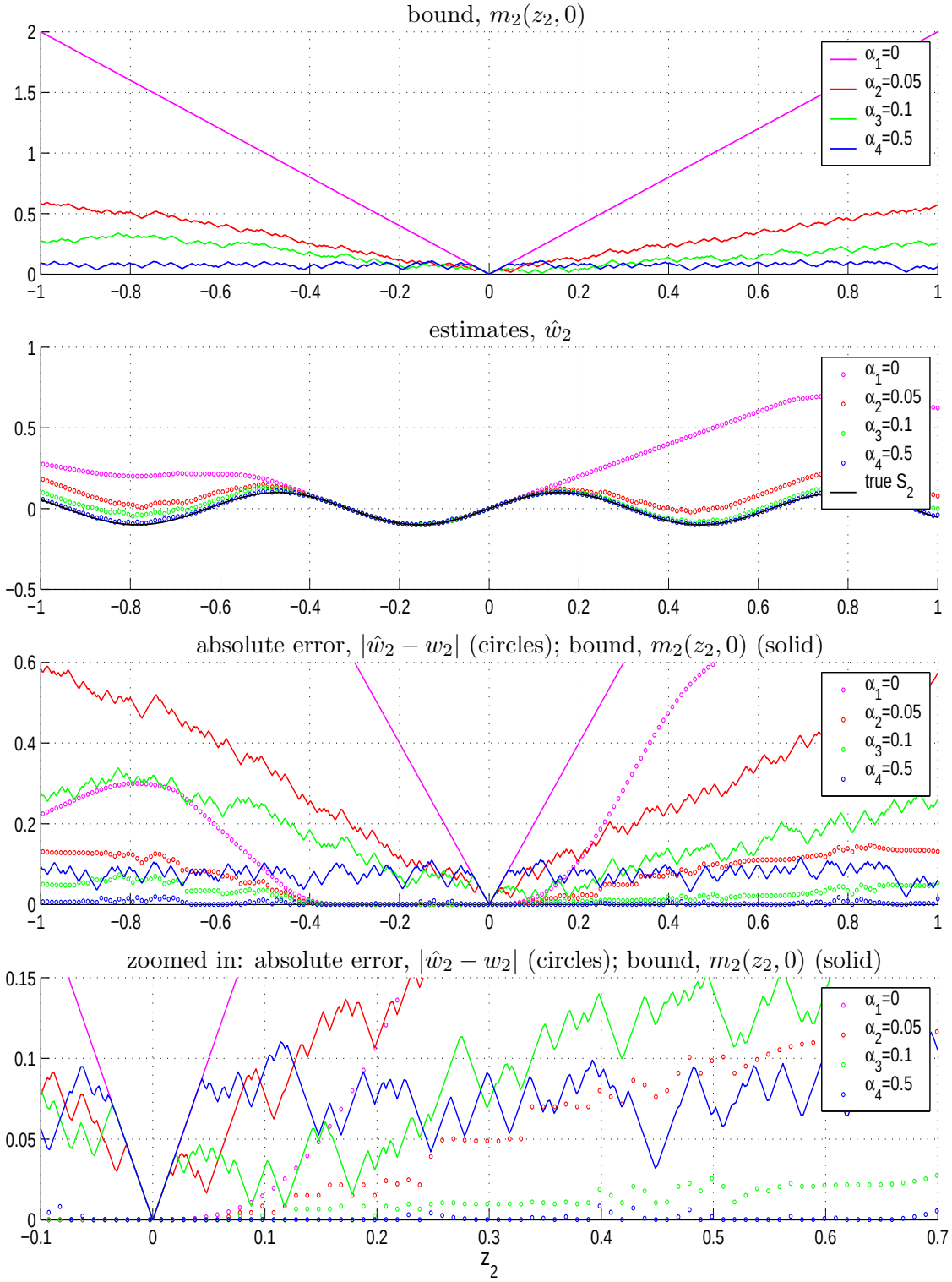


Figure 5.8: Example 2: Bounds and formulation A estimates of $\mathcal{S}_2$, for the four inputs.

increasing z_1 (and since this is a static system the ordering in time of the data record is not important). Then z_2 would appear to have the jitter, increasing in variance from α_1 to α_4.

The second estimation formulation does not impose the Lipschitz constraints, in order to show that the characteristics of input records that we have found to be important when these constraints are assumed are still relevant for other related estimation formulation where the assumptions are not met necessarily. The second estimation scheme is to minimize the dispersion , subject this time only to the $\mathcal{S}_1(0) = 0.8$ constraint:

$$\hat{w} = \begin{array}{ll} \underset{w}{\text{argmin}} & \|D^1 P^{z_1} w_1\|_2^2 + \|D^1 P^{z_2} w_2\|_2^2 \\ \text{subject to} & y(t) = w_1(t) + w_2(t) \quad \forall t \in T \\ & \mathcal{S}_1(0) = 0.8. \end{array} \tag{B}$$

Estimation results for the first nonlinearity, using the four data records, are shown in Figure 5.9, with the previously-computed m_1 bounds. The major trends are similar. It so happens the bounds are still met, but there are no guarantees for this estimation formulation.

Finally in this section, consider a system that is slightly more general than the one we have been considering up until now:

$$y(t) = \theta_1 \mathcal{S}_1(z_1(t)) + \theta_2 \mathcal{S}_2(z_2(t)). \tag{5.25}$$

As compared to (5.1), in (5.25) we allow that the weightings in the output equation of the contributions from the individual functions may be different than unity. What effect might this have on bounds and estimates? As the system is still fairly simple we can isolate this effect and compare results for different choices of weights.

It is easy to adjust the bounding scheme of Section 5.1 to account for the added generality (this is a preview of the more general process presented in the next section). Here, (5.5) becomes

$$\theta_1 d_1\big(z_1(t)\big) + \theta_2 d_2\big(z_2(t)\big) = 0, \quad \forall t \in T, \tag{5.26}$$

so that

$$d_1(z_1(t)) = -\frac{\theta_2}{\theta_1} d_2(z_2(t)), \tag{5.27}$$

and in forming $k_1(\cdot,\cdot)$, (5.12) becomes

$$\Big| d_1(z_1(t)) - d_1(z_1(s)) \Big| \leq \min \Big\{ 2\gamma_1 |z_1(t) - z_1(s)| \ , \ \Big| \frac{\theta_2}{\theta_1} \Big| 2\gamma_2 |z_2(t) - z_2(s)| \Big\} \quad \forall t, s \in T. \tag{5.28}$$

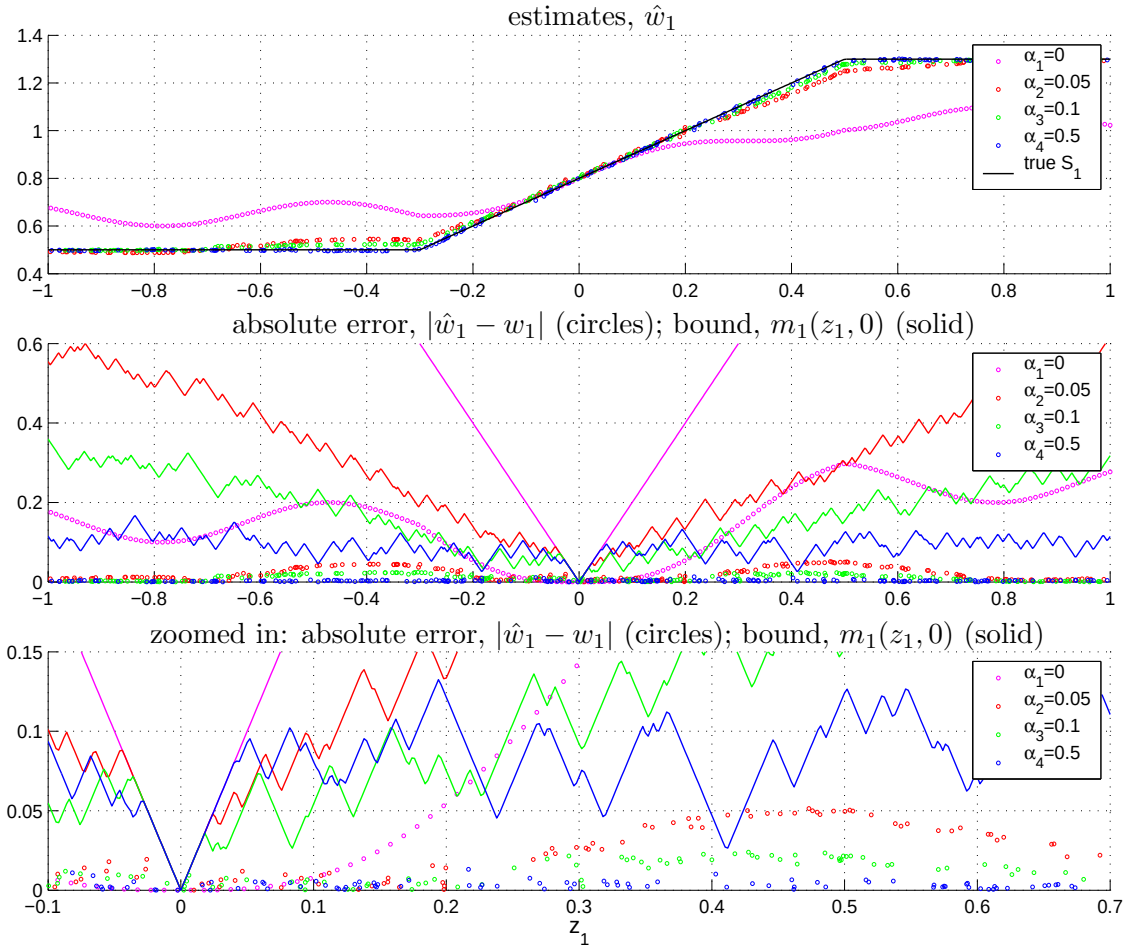


Figure 5.9: Example 2: Bounds and formulation B estimates of $\mathcal{S}_1$, for the four inputs.

Comparing these two, we see that the smaller $|\theta_2/\theta_1|$, the smaller we expect $k_1(\cdot,\cdot)$ to be. The situation is symmetric in the two functions, so that

$$\Big|d_2(z_2(t)) - d_2(z_2(s))\Big| \leq \min\left\{ \left|\frac{\theta_1}{\theta_2}\right| 2\gamma_1|z_1(t) - z_1(s)| \ , \ 2\gamma_2|z_2(t) - z_2(s)| \right\} \ \ \forall t, s \in T, \quad (5.29)$$

and the smaller $|\theta_2/\theta_1|$, the larger we expect $k_2(\cdot,\cdot)$ to be. In other words, the larger the relative magnitude of the contribution of the output of $\mathcal{S}_i$ in the output y, the smaller the bounds on the estimation error for that function (and so, we infer, the better the estimates). This makes some intuitive sense, in that as functions have a lesser effect on the output, or play a less important role in the system's input-output behavior, it becomes harder to know what they are and to form estimates.

In "Example 3" we consider bounds and estimates with system (5.25) for three choices of weights, with $[\theta_1 \ \ \theta_2]$ equal to $[1 \ \ 1]$ (as in Example 2), $[4 \ \ 1]$, and $[1 \ \ 4]$ respectively.

In each case we use the same input as the $\alpha_2 = 0.05$ input of Example 2—see (5.24). We also use the same two estimation formulations, A and B, as in that example. Using the modified procedure for computing them, the bounds m_1 and m_2 are shown in Figure 5.10, along with the formulation A estimates, for the three systems. The formulation B estimates, sans bounds, are shown in Figure 5.11. In these figures we indeed see that as the contribution of the output of $\mathcal{S}_1$ in the output becomes relatively larger, the bound m_1 becomes smaller and the estimate $\hat{\mathcal{S}}_1$ becomes more accurate, under formulation A as well as formulation B. And conversely the bound m_2 grows larger and the estimate $\hat{\mathcal{S}}_2$ degrades.

5.3 General Formulation

In Section 5.1 and 5.2, in presenting the main ideas of this chapter, we considered exclusively the static, sum-of-2 nonlinearities system as in (5.1) or (5.25). In this section we present ideas for extending the bounding approach to a significantly more general system, including dynamics, multiple outputs, more than 2 nonlinearities, and noise, ones which can be put in the following form:

$$y(t) = \sum_{k=0}^{\infty} F(k)y(t-k) + \sum_{k=0}^{\infty} E(k)u(t-k) + \sum_{k=0}^{\infty} G(k)w(t-k) + \sum_{k=0}^{\infty} H(k)e(t-k), \quad (5.30)$$

and

$$w_i(t) = \mathcal{S}_i(z_{[i]}(t)). \quad (5.31)$$

Here we no longer restrict $\mathcal{S}_i$ to be single-input (as in Section 3.3.5, by the notation $z_{[i]}$ we mean that subset of the n_z scalar signals comprising z that w_i is a function of). To form bounds we use knowledge of the input z to $\mathcal{S}$, in other words z is treated as a known input to the system. As well, u and y are measured, while e is an unmeasured disturbance. Note that system structure is a subset of Wiener nonlinear systems, the subset where the linear part is a so-called ARMAX system. Although w and e are not assumed to be known, estimates of them satisfy the linear system equations so that

$$y(t) = \sum_{k=0}^{\infty} F(k)y(t-k) + \sum_{k=0}^{\infty} E(k)u(t-k) + \sum_{k=0}^{\infty} G(k)\hat{w}(t-k) + \sum_{k=0}^{\infty} H(k)\hat{e}(t-k). \quad (5.32)$$

subtracting (5.30) from (5.32) we have

$$0 = \sum_{k=0}^{\infty} G(k)d(z(t-k)) + \sum_{k=0}^{\infty} H(k)f(z(t-k)), \quad (5.33)$$

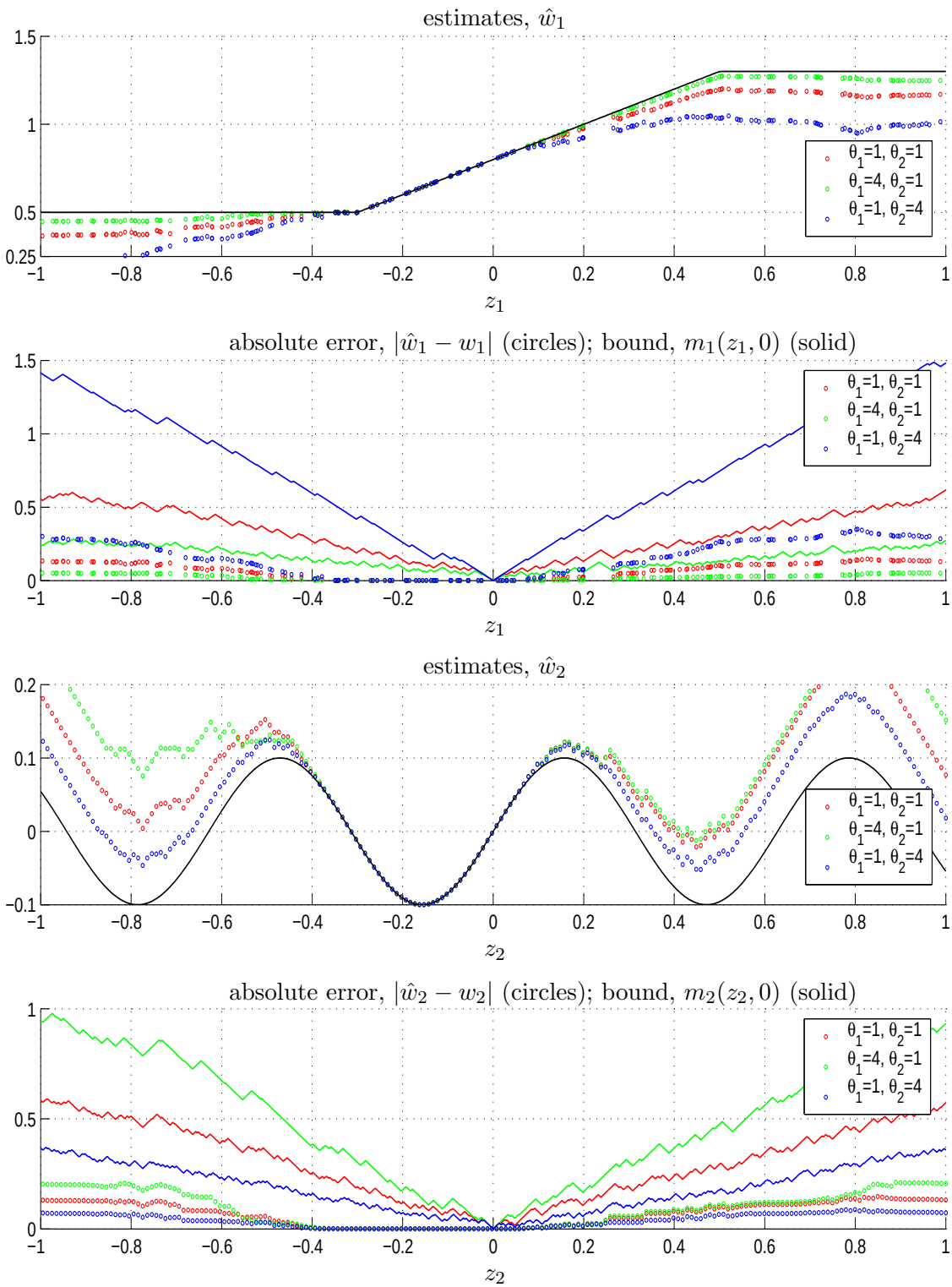


Figure 5.10: Example 3: Bounds and formulation A estimates of $\mathcal{S}_1$ and $\mathcal{S}_2$, for three different $G(0)$.

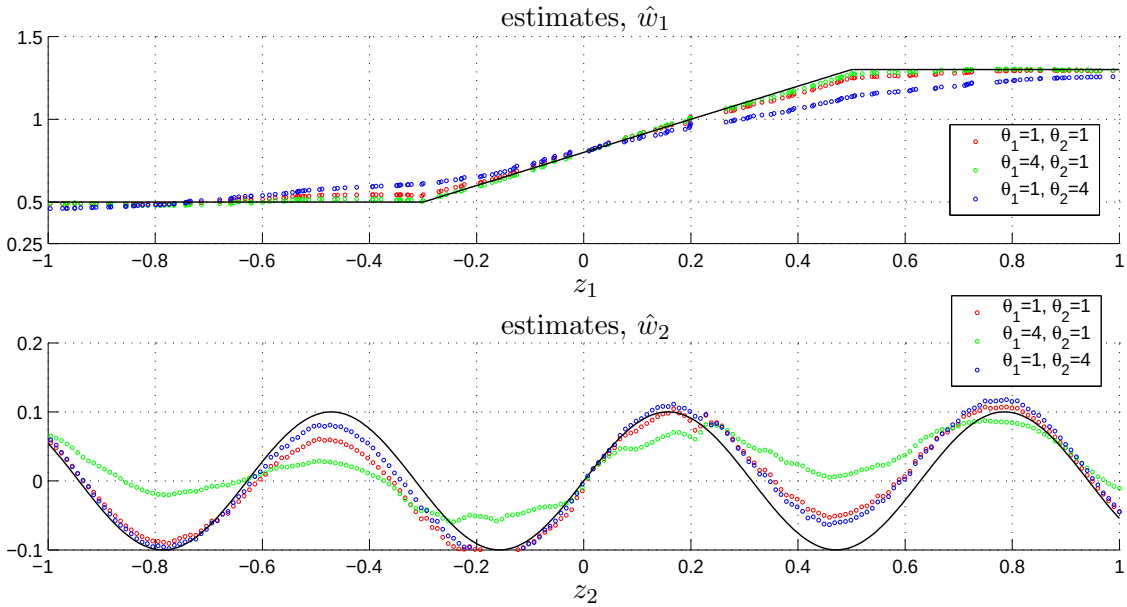


Figure 5.11: Example 3: Bounds and formulation B estimates of $\mathcal{S}_1$ and $\mathcal{S}_2$, for three different $G(0)$.

where

$$d(z(t)) := \hat{\mathcal{S}}(z(t)) - \mathcal{S}(z(t)), \qquad \text{or} \qquad \begin{bmatrix} d_1(z_{(1)}(t)) \\ \vdots \\ d_{n_w}(z_{(n_w)}(t)) \end{bmatrix} = \begin{bmatrix} \hat{\mathcal{S}}(z_{(1)}(t)) - \mathcal{S}(z_{(1)}(t)) \\ \vdots \\ \hat{\mathcal{S}}_{n_w}(z_{(n_w)}(t)) - \mathcal{S}_{n_w}(z_{(n_w)}(t)) \end{bmatrix} \tag{5.34}$$

and

$$f(t) := \hat{e}(t) - e(t) \tag{5.35}$$

and as before we form estimates using

$$\hat{\mathcal{S}}(z(t)) := \hat{w}(t), \quad t \in T \tag{5.36}$$

so that on $t \in T$

$$\hat{w}(t) - w(t) = d(z(t)) \tag{5.37}$$

Fixing our attention now on the J'th function, consider forming a bound on the difference in d_J between pairs of points in its domain. As before we use the data record, known linear system, and Lipschitz assumptions to form $k_J(\cdot,\cdot)$, an initial bound between the pairs of points in the input data record. In Section 5.1, Equation (5.12), we formed $k_1(z_1(t), z_1(s))$ as the minimum of a the direct Lipschitz bound, and a bound arrived at

using the system equations and assumptions about the estimate. The situation is the same here, except that there are potentially more candidates of the latter type included in the minimization. For every nonzero element in the J'th column of any of the $G(k)$, we can use (5.33) to form a bound on $|d_J(z_{(J)}(t)) - d_J(z_{(J)}(s))|$ Considering the i'th row (which corresponds to the i'th output) of (5.33) at two times t and s and subtracting these, we have

$$\sum_{\substack{k=0..\infty \\ j=1..n_w}} G_{ij}(k)\Big[d_j(z_{(j)}(t-k)) - d_j(z_{(j)}(s-k))\Big] \; + \sum_{\substack{k=0..\infty \\ j=1..n_e}} H_{ij}(k)\Big[f_j(t-k) - f_j(s-k)\Big] \; = \; 0 \tag{5.38}$$

for all $i = 1..n_y$ and all $(t, s) \in T$. Now for any (i, K) such that $G_{iJ}(K) \neq 0$, by isolating that term and using the triangle inequality we get the bound

$$\Big|d_J(z_{(J)}(t-K)) - d_J(z_{(J)}(s-K))\Big| \leq \frac{1}{|G_{iJ}(K)|}\Bigg\{ \sum_{\substack{k=0..\infty \\ j=1..n_w \\ (k,j)\neq(K,J)}} |G_{ij}(k)|\Big|d_j(z_{(j)}(t-k)) - d_j(z_{(j)}(s-k))\Big| + \sum_{\substack{k=0..\infty \\ j=1..n_e}} |H_{ij}(k)|\Big|f_j(t-k) - f_j(s-k)\Big|\Bigg\} \tag{5.39}$$

As before we assume the functions satisfy Lipschitz conditions, i.e. $\mathcal{S}_j \in L_{\gamma_j}$, and also that estimates meet the Lipschitz constraints, i.e.

$$|\hat{w}_j(t) - \hat{w}_j(s)| \leq \gamma_j \|z_{(j)}(t) - z_{(j)}(s)\|. \tag{5.40}$$

Therefore we have, for $j = 1..n_w$,

$$\begin{aligned} |d_j(z_{(j)}(t)) - d_j(z_{(j)}(s))| &= \Big|\{\hat{\mathcal{S}}_j(z_{(j)}(t)) - \mathcal{S}_j(z_{(j)}(t))\} - \{\hat{\mathcal{S}}_j(z_{(j)}(s)) - \mathcal{S}_j(z_{(j)}(s))\}\Big| \\ &= \Big|\{\hat{\mathcal{S}}_j(z_{(j)}(t)) - \hat{\mathcal{S}}_j(z_{(j)}(s))\} - \{\mathcal{S}_j(z_{(j)}(t)) - \mathcal{S}_j(z_{(j)}(s))\}\Big| \\ &\leq 2\gamma_j \|z_{(j)}(t) - z_{(j)}(s)\|. \end{aligned} \tag{5.41}$$

This bounds some of the quantities in (5.39). We need to also bound

$$|f_j(t) - f_j(s)| = \Big|\{\hat{e}_j(t) - e_j(t)\} - \{\hat{e}_j(s) - e_j(s)\}\Big| \tag{5.42}$$

But this expression involves stochastic quantities and it is less clear how to proceed in bounding it. If e_j is IID then we have no information about how $e_j(t)$ might be related

to $e_j(s)$. This is unlike the case for d_j, where we have some information about how the relationship between $d_j(z_{(j)}(t))$ at different times because of the Lipschitz information. This was not an issue in the previous section because the system there did not have a stochastic disturbance input. If our stochastic model for e_j is an IID uniformly distributed process, say $e_j(t) \sim U[a_j, b_j]$, then the following is valid

$$|e_j(t) - e_j(s)| \leq b_j - a_j \tag{5.43}$$

and in the estimation formulation we could constrain $\hat{e}_j(t) \in [a_j, b_j]$ and it follows that

$$|\hat{e}_j(t) - \hat{e}_j(s)| \leq b_j - a_j \tag{5.44}$$

and putting it together

$$|f_j(t) - f_j(s)| \leq 2(b_j - a_j). \tag{5.45}$$

However if the stochastic model for e_j is an IID normal process then there is no finite deterministic bound on $|e_j(t) - e_j(s)|$, since for any N the probability that $|e_j(t) - e_j(s)| > N$ is nonzero.

The fundamental issue is that we're looking for deterministic bounds in a situation involving stochastic quantities. A more appropriate question would be something like, "what is the probability that the absolute value of the error, or differences in error, exceeds such-and-such?". However in the present development we are not employing a stochastic framework to answer questions like this; this may be a useful extension to consider at some point. A possible work-around is to pick a value such that $|e_j(t) - e_j(s)|$ exceeds it with only a small probability, and be satisfied that the bound will be valid in a high percentage of experimental noise realizations. If we are satisfied with β_j for this value, and say we also enforce or think with high probability that

$$|\hat{e}_j(t) - \hat{e}_j(s)| \leq \beta_j \tag{5.46}$$

then with acceptably high probability

$$|f_j(t) - f_j(s)| \leq 2\beta_j \tag{5.47}$$

Equation (5.39) becomes

$$\Big|d_J(z_{(J)}(t-K)) - d_J(z_{(J)}(s-K))\Big| \leq$$
$$\frac{1}{|G_{iJ}(K)|}\left\{ \sum_{\substack{k=0..\infty \\ j=1..n_w \\ (k,j)\neq(K,J)}} |G_{ij}(k)|2\gamma_j \Big|z_{(j)}(t-k) - z_{(j)}(s-k)\Big| + \sum_{\substack{k=0..\infty \\ j=1..n_e}} |H_{ij}(k)|2\beta_j \right\} \quad (5.48)$$

The RHS here is all known quantities and can therefore be used for bounding. Now to form $k_J(z_{(J)}(\tilde{t}), z_{(J)}(\tilde{s}))$, the initial bound on $|d_J(z_{(J)}(\tilde{t})) - d_J(z_{(J)}(\tilde{s}))|$, analogously to what we did previously in (5.12), we pick the smallest number as the RHS of (5.48) varies over all (outputs) i, (lags) K, and (times) t and s such that $t - K = \tilde{t}$, $s - K = \tilde{s}$, $G_{iJ}(K) \neq 0$. Also include in this minimization the direct Lipschitz bound,

$$|d_J(z_{(J)}(\tilde{t})) - d_J(z_{(J)}(\tilde{s}))| \leq 2\gamma_J \|z_{(J)}(\tilde{t}) - z_{(J)}(\tilde{s})\| \quad (5.49)$$

With k_J formed, it is straightforward to apply the next step, the all-pairs shortest-path graph search minimization, to form l_J. Once again the graph has L nodes, one for each time present in the data record. For multi-input $\mathcal{S}_J$ the next step, of forming m_J, the extension of the bound to arbitrary pairs of points in D_J, is not as straightforward as before. This is because for a given pair it's a more involved geometrical problem to decide which points in the data record $Z_{(J)}$ of $z_{(J)}$ we need to check as we did in (5.17). Regardless, l_J is still valid, and the parenthesized alternative suggested on 148 is a possible way to attack the problem. Of course, for multi-input functions the computed bound is also harder to visualize. For single-input $\mathcal{S}_J$ these issues goes away.

In the discussion of the k bound following Equation (5.12) we noted that the bound due to terms like (5.48) can be small even when the direct Lipschitz bound is large and this effect is primarily responsible for bounds that ultimately beat the direct Lipschitz bound. With noise, or at least with the way suggested above for dealing with noise signals, the situation may not be quite as good. Note that the term on the RHS of (5.48) is always at least as big as

$$\sum_{\substack{k=0..\infty \\ j=1..n_e}} |H_{ij}(k)|2\beta_j \quad (5.50)$$

which is a constant independent of the input signal and pair (s, t) of times. This is a lower limit on how small the bound can get, and is a consequence of putting deterministic

bounds on stochastic quantities. The hope is this number is small relative to the sizes of nonlinearities e.g. with noise levels that are known to be low.

In Section 5.1 we saw that knowledge of the system equation (5.1), a data record, and smoothness constraints is not enough to rule out constant translations when forming estimates of $\hat{\mathcal{S}}_1$ and $\hat{\mathcal{S}}_2$. In essence $d_1 = c$ and $d_2 = -c$ is a function that lies in the nullspace of (5.5), and being constant escapes the scrutiny of Lipschitz or any other smoothness constraints as well as restrictions due to one of the z_i varying while the other does not. To put it another way we cannot, without additional information about the value of either $\mathcal{S}_1$ or $\mathcal{S}_2$ at some point in its domain, distinguish between

$$\mathcal{S}(z) \quad \text{and} \quad \mathcal{S}(z) + v \tag{5.51}$$

where v is any element of the space

$$V^\dagger = \left\{ \begin{bmatrix} 1 \\ -1 \end{bmatrix} c : c \in \mathbf{R} \right\} \tag{5.52}$$

With the more general system (5.30) the same ideas hold. Smoothness criteria cannot distinguish between constant translations. In this case the set of constant translations of $\mathcal{S}(z)$ that are consistent with data records (5.30) is the space

$$V = \bigcap_{k=0}^{\infty} \mathcal{N}(G(k)), \tag{5.53}$$

vectors that lie in the nullspaces of all of the matrices $G(k)$, because for $d(\cdot) = v \in V$ the first term in the sum of (5.33) is zero. Note for the sum-of-2 system we had $G(0) = [1 \quad 1]$ and $G(k) = 0$ for $k > 0$, and so $V = \mathcal{N}([1 \quad 1]) = V^\dagger$. As before we can either treat such constant translations of $\mathcal{S}$ as acceptable estimates, or require enough additional information about the value of $\mathcal{S}$ at some points in its domain to resolve the particular element of the "estimate space", i.e. the constant translations, that is the correct one.

We note a couple general trends with the bounds. First, the more outputs the system has, the better (smaller) the bound will be. This is because for each s, t pair there are more choices in the minimization that forms $k_J(z_{(J)}(s), z_{(J)}(t))$. This is as we would hope, that an experiment that contains more information, for instance by using additional sensors, can provide better estimates. Secondly, the more nonlinearities involved the worse (larger) the bounds. This is because in order for (5.48) to be small we need the points

$z_{(j)}(t-k)$ and $z_{(j)}(s-k)$ to be small for all $j = 1..n_w$, $j \neq J$, simultaneously. For large n_w it happens less frequently in the data record that all $n_w - 1$ of these pairs are close, and a larger data record length is needed to achieve the same bound. The trends we noted in the previous section still hold, namely those regarding data density, the relative variance of $z_{(J)}$ with respect to the other functions' inputs, and relative weighting of a given function in the output (as measured by the size of the elements of G).

Thus far the bounds on estimation error, and the computation of them, does not depend on computing any estimates. The bounds are functions primarily of the linear part of the system, the inputs to the unknown functions, assumed Lipschitz constants for these functions, and properties that estimates possess in general. This is an attractive situation because the process can reveal how bounds, and the quality of estimates, depend on these factors, as we have been discussing. However, given actual estimates $\hat{w}$ and $\hat{e}$, we can use these to refine bounds, or make them less conservative. This is because in bounding expressions like (5.39) we can replace bounds on quantities like $|\hat{\mathcal{S}}_j(z_{(j)}(t)) - \hat{\mathcal{S}}_j(z_{(j)}(s))|$ and $|\hat{e}_j(t) - \hat{e}_j(s)|$ with their actual values. This type of aposteriori bound is more oriented toward bounding the quality of a particular estimate, as opposed to exploring how the potential quality of estimates depends on various factors. It is also useful when Lipschitz bounds on estimates are not known beforehand, or when the properties of noise estimates is unclear.

5.4 Review

To summarize, in this chapter we presented an approach to bounding the estimation error, for a class of problems involving identification of static nonlinearities. In deriving these bounds we assumed that estimates have several properties, including consistency with an assumed linear system structure and a Lipschitz smoothness constraint. In particular, these properties are natural consequences of the estimation procedures we proposed in earlier chapters for such identification problems. In addition it was assumed that the input z to the unknown functions is available, as it is used in the bounding computation. It is an open problem as to how we might compute analogous bounds in estimation problems where the z signal is not fully known. This is consistent with a common theme in this dissertation, that availability of z leads to formulations that are much more tractable than when this signal is unknown.

While certainly valid, in examples we have seen that the bounds can be conservative. For a given set of computed estimates we can reduce the conservatism by replacing overbounds of various expressions involving estimates with the exact value. At any rate, for a class of estimation formulations we can compute bounds on the errors.

Another application of the bounding ideas is to test and compare different input signals, for a given model structure and in advance of collecting data, in order to guide experiment design, i.e. the problem of determining what input characteristics are important in order to generate experimental data that yields acceptable estimates. We indicated several such properties, including density on the domain of the unknown function, independent variation of the inputs to the function being bounded, and length of the data record. For these cases we argued intuitively how the property should affect bounds, and saw this behavior in examples. We also showed that the general trends seen in the bounds carry over to when actual estimates are computed, both for formulations that satisfy the assumptions used to derive bounds, and similar types of formulations that do not necessarily meet the assumptions.

A final benefit of thinking about bounds is insights gained about what various characteristics of model structure imply for the estimation problem. Some specific characteristics that we looked at were the number of unknown functions, the number of outputs, noise levels, weightings of the individual $\mathcal{S}_i$ in the output, and Lipschitz property of the $\mathcal{S}_i$. When things don't seem to be working well, we can now point to at least a few things to check.

Chapter 6

Conclusions

This thesis contains ideas for estimating unknown static functions within interconnections made up of linear systems and static operators. The approach considered here involves using experimental input-output data and knowledge about the system to form estimates of the inputs and outputs of the unknown functions that occurred during the experiment. Another interpretation of the problem being solved is that of signal estimation under this condition of having knowledge that certain elements of the system are static operators. This approach is fundamentally nonparametric.

The basic idea is straightforward. We may not know or have access to the signals we'd like to have, and so we endeavor to form estimates of the unknown signals in the problem, taking advantage of the information that is available. Some signals in the problem *are* measured, and these are related to the uncertain signals through a linear system in a known way. This represents a constraint on the unknowns. Another constraint is that certain of the signals are related to each other in a static way, and a third is that estimates of signals modeled as stochastic processes should possess properties that are consistent with the stochastic model.

The sizes of the estimation problems we form are large, because they involve estimating signals, and lengths of data records can be long. Because of this care must be taken to form problems that are actually solvable, and fit into optimization frameworks that can handle a large number of decision variables and constraints. This is where much of the time has been spent. Firstly, the linear system represents a large system of equality constraints. With the right approach these can be handled in a recursive, efficient way, and effectively taken out of the problem. Detailed algorithms for doing this were given in

Section 3.2.

Next, devising and studying efficient measures of staticness was a main emphasis of Chapter 3, and indeed a main emphasis of the thesis. We found that the dispersion measure strikes a reasonable balance between simplicity and effectiveness. Several closely related measures were also suggested, and different combinations of these may prove useful depending on the specifics of the problem being solved. We examined, in a stochastic setting, how good a job the dispersion operator does at distinguishing between signals related by a static operator and those that are not. At a more basic level, we also discussed how the dispersion operator agrees with our intuition for recognizing static relationships.

Lastly we discussed a few simple considerations in order to make sure that estimates of noise signals are consistent with any assumed stochastic properties of them.

We saw time and again that the staticness measures become difficult objects to optimize when z is not known, that is, when z depends on the decision variables of the problem. This is a major stumbling block for implementation. An iterative method to get around this problem is suggested, and shown to produce reasonable results in several examples. However this so-called bootstrap method also has its shortcomings. For instance in one example we saw that the iteration did not find optimal solutions of the original problem posed, nor does it produce a monotonic sequence of estimates (which might be considered desirable). Perhaps there are refinements or alternatives to the method that do not suffer these shortcomings. Along these lines, another thing to be investigated is whether slightly different formulations of the main problem we used, using different combinations of the optimization elements listed throughout, might reliably produce better results. Yet another is to try out some well-known methods for using convex solver to find approximate solutions to nonconvex problems.

Finally in Chapter 5 we presented computational methods to bound the degree of mismatch between estimates and the true function, under assumptions on the unknown functions and estimation procedure that are characteristic of some of the estimation methods in this thesis. These computations are based on the input to the experiment, and so can say something about the potential a specific input signal has to produce favorable estimation results.

It is important to keep in mind how the methods used in this thesis might fit into a larger picture of estimation or system identification. One major assumption is that the linear part of the system is exactly known. This is not a technically restrictive assumption,

in that any model structure can be put into the correct form. However it *is* practically restrictive, in that the more of the system can be pulled out into the known linear part, the more tractable a solution becomes.

The methods are not, however, compatible with known *nonlinear* parts of the model. This is because any attempt to impose a nonlinear constraint between signals immediately renders a problem nonconvex, and therefore outside the scope of what we consider solvable for these variable sizes. This is a potential area for future investigation, to see if there is any progress to be made with respect to being more amenable to using different types of prior information.

We think these methods could be potentially useful in finding starting points for nonlinear solvers in parameter estimation problems. In general we could characterize the estimation methods under consideration as rough, but robust. They seem to perform well and degrade gracefully under a wide variety of situations. On the other hand the final estimates that are produced often do not exactly match the true functions, nor do they seem to approach the exact nonlinearity in an asymptotic sense. On the other hand parameter estimation methods, which are impressive when they converge, are also subject to being sensitive to choice of initial estimate and basis choice, and can fail spectacularly. A possibility is to use the convex programming methods to find a good starting point, and then let general nonlinear optimizers clean up that estimate.

Bibliography

[1] S. Billings. Identification of nonlinear systems–a survey. *IEE Proceedings D (Control Theory and Applications*, 127(6):272–85, Nov. 1980.

[2] S. A. Billings and S. Y. Fakhouri. Identification of systems containing linear dynamic and static nonlinear elements. *Automatica*, 18(1):15–26, 1982.

[3] S. A. Billings and L. M. Li. Continuous time nonlinear system identification in the frequency domain. *International Journal of Control*, 74(11):1052–61, 2001.

[4] S. A. Billings and W. S. F. Voon. Structure detection and model validity tests in the identification of nonlinear systems. *IEE Proceedings D (Control Theory and Applications)*, 130(4):193–9, 1983.

[5] S. Boyd and L. O. Chua. Fading memory and the problem of approximating nonlinear operators with volterra series. *IEEE Transactions On Circuits and Systems*, 32(11): 1150–1161, 1985.

[6] S. Boyd, L. El Ghaoui, E. Feron, and V. Blakrishnan. *Linear matrix inequalities in system and control theory*. SIAM studies in applied mathematics, volume 15, 1994.

[7] D. R. Brillinger. The identification of a particular nonlinear time series system. *Biometrika*, 64:509–515, 1977.

[8] C. Chen and S. Fassois. Maximum likelihood identification of stochastic wiener-hammerstein-type non-linear systems. *Mechanical Systems and Signal Processing*, 6 (2):135–53, Mar. 1992.

[9] S. Chen and S. Billings. Representations of nonlinear systems: the narmax model. *International Journal of Control*, 49(3):1013–32, Mar. 1989.

[10] K. Chernyshov and F. Pashchenko. Mildly formalized system identification based on consistent measures of dependence. In *Proceedings of the 16th IEEE Instrumentation and Measurement Technology Conference*, volume 2, pages 904–9, May 1999.

[11] M. Claassen. *System Identification for Structured Nonlinear Systems.* PhD thesis, University of Caliornia, Berkeley, 2001.

[12] M. Frechet. Sur les fonctionelles continues. *Ann.ł'Ecole Normale Sup.*, 27, 1910.

[13] A. S. French and E. G. Butz. Measuring the wiener kernels of a non-linear system using the fast fourier transform algorithm. *International Journal of Control*, 17(3):529–39, 1973.

[14] P. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, paperback, 1st edition, 1981.

[15] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinate programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[16] W. Greblicki and M. Pawlak. Cascade non-linear system identification by a nonparametric method. *International Journal of Systems Science*, 25(1):129–53, 1994.

[17] R. Haber and L. Keviczky. Identification of nonlinear dynamic systems. In *Identification and System Parameter Estimation: Proceedings of the 4th IFAC Symposium*, pages 79–126, 1978.

[18] W. Härdle, Lütkepohl, and R. Chen. A review of nonparametric time series analysis. *International Statistical Review*, 65(1):49–72, 1997.

[19] G. Hung and L. Stark. The kernel identification method (1910-1977)-review of theory, calculation, application and interpretation. *Mathematical biosciences*, 37(3–4):135–90, 1977.

[20] G. Hung, L. Stark, and P. Eykhoff. On the interpretation of kernels. i. computer simulation of responses to impulse pairs. *Annals of biomedical engineering*, 5(2):130–43, 1977.

[21] T. Johansen. On tikhonov regularization, bias and variance in nonlinear system identification. *Automatica*, 33(3):441–6, 1997.

[22] R. Johansson. *System Modeling and Identification*. Prentice Hall, 1993.

[23] A. Juditsky, H. kan Hjalmarsson, A. Benveniste, B. Delyon, L. Ljung, J. Sjöberg, and Q. Zhang. Nonlinear black-box models in system identification: Mathematical foundations. *Automatica*, 31(12):1752–1750, 1995.

[24] S. L. Lacy and D. S. Bernstein. Identification of FIR wiener systems with unknown, noninvertible, polynomial nonlinearities. 2:893–8, 2002.

[25] L. Ljung. *System Identification: Theory for the User*. Prentice Hall PTR, 2nd edition, 1999.

[26] L. Ljung. *System Identification Toolbox–User's Guide*. The MathWorks, 2000.

[27] Luenberger. *Linear and Nonlinear Programming*. publisher, 1984.

[28] D. G. Luenberger. *Optimization by Vector Space Methods*. Wiley, 1969.

[29] K. Narendra and P. Gallman. An iterative method for the identification of nonlinear systems using the hammerstein model. *IEEE Transactions On Automatic Control*, 11 (7):546–50, July 1966.

[30] B. Noble and J. Daniel. *Applied Linear Algebra*. Prentice Hall, 3rd edition, 1988.

[31] F. Paganini. A set-based approach for white noise modeling. *IEEE Transactions on Automatic Control*, 41(10):1453–65, October 1996.

[32] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice Hall, 1982.

[33] R. K. Pearson. Nonlinear input/output modelling. *Journal of Process Control*, 5(4): 197–211, 1995.

[34] S. Ross. *Introduction to Probability and Statistics for Engineers and Scientists*. Wiley, 1987.

[35] M. Schetzen. *The Volterra and Wiener Theories of Nonlinear Systems*. John Wiley & Sons, 1980.

[36] SeDuMi. `http://fewcal.kub.nl/sturm/software/sedumi.html`. a Matlab toolbox for optimization over symmetric cones. by Jos F. Sturm.

[37] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications*. Springer, 2000.

[38] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. kan Hjalmarsson, and A. Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724, 1995.

[39] T. Söderström and P. Stoica. *System Identification*. Prentice Hall, 1989.

[40] H. Unbehauen and G. P. Rao. A review of identification in continuous-time systems. *Annual Reviews in Control*, 22:145–71, 1998.

[41] L. Vandenberghe and S. Boyd. Semidefinate programming. *SIAM Review*, 38(1):49–95, 1996.

[42] G. Vandersteen and J. Schoukens. Measurement and identification of nonlinear systems consisting out of linear dynamic blocks and one static nonlinearity. In *IEEE Instrumentation and Measurement Technology Conference. Sensing, Processing, Networking.*, volume 2, pages 853–8, Ottawa, Ontario, Canada, 1997.

[43] E. Wemhoff. A nonparametric method for identification of a static nonlinearity in a structured-nonlinear system. Master's thesis, University of California, Berkeley, 1998.

[44] E. Wemhoff, A. Packard, and K. Poolla. On the identification of nonlinear maps in a general interconnected system. In *Proceedings of the 1999 American Controls Conference*, volume 5, pages 3456–61, 1999.

[45] D. Westwick and M. Verhaegen. Identifying mimo wiener systems using subspace model identification methods. *Signal Processing*, 52(2):235–58, 1996.

[46] K. Zhou, J. C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, 1996.