Abstract

Drake, Kimberly J. Analysis of Numerical Methods for Fault Detection and Model Identification in Linear Systems with Delays. (Under the direction of Steve Campbell.)

Recently an approach for multi-model identification and failure detection in the presence of bounded energy noise over finite time intervals has been introduced. This approach involved offline computation of an auxiliary signal and online application of a hyperplane test.

This approach has several advantages; but, as presented, observation over the full time interval was required before a decision could be made. We develop an algorithm which modifies this approach to permit early decision making with the hyperplane test.

In addition, we extend this approach to handle problems that include delays. The original method requires the formulation and solution of an optimal control problem. We approach these problems in three ways. The first is through the Method of Steps, reformulating the system without delays so that we might apply existing theory with modifications. Also, we approximate the delayed systems using splines and central differences, eliminating the delay so that existing theory will apply. Approximations allow for more complicated models than the Method of Steps; however, the Method of Steps is a true solution, rather than an approximate one. Thus, solutions using the Method of Steps serve as a basis of comparison and verification of the approximate methods.

ANALYSIS OF NUMERICAL METHODS FOR FAULT DETECTION AND MODEL IDENTIFICATION IN LINEAR SYSTEMS WITH DELAYS

BY

KIMBERLY J. DRAKE

A DISSERTATION SUBMITTED TO THE GRADUATE FACULTY OF

NORTH CAROLINA STATE UNIVERSITY

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

APPLIED MATHEMATICS

RALEIGH, NORTH CAROLINA

AUGUST 22, 2003

APPROVED BY:

For my beloved niece, Kaitlyn,

One day when you were a very, very little girl, you put on the apron and vest from my waitressing uniform and you played at taking orders from everyone in the house, including your dolls. You were so sweet, sassy and small, going from person to doll with equal gravity, taking orders in a language the rest of the family was too old to remember. As I watched you, I realized that you were not just playing at being a waitress; you were playing at being me. I realized that day how very important it is that I be a woman who you could become. You are my inspiration.

Always remember that *nothing* is beyond your reach.

Biography

Kimberly J. Drake was born in Dover, NJ on January 7, 1970. She grew up in the small town of Budd Lake, NJ and graduated in 1988 from Mt. Olive High School. She worked her way through college, graduating from Montclair State University in 1996 with a B.S. in mathematics and computer science, as well as a teaching certificate.

After college, she moved briefly to Livermore, CA where she interned at the Lawrence Livermore National Laboratory. Leaving sunny California behind, Kimberly moved to Raleigh, NC where she attended graduate school at North Carolina State University, working with Steve Campbell.

Her graduate school years included a time working at the Boeing Company with John Betts in Bellevue, WA and a time working at INRIA-Rocquencourt with Ramine Nikoukhah in Le Chesnay Cedex, France. After graduating with her Ph.D. in Applied Math, she plans to move to Philadelphia, PA to work for the Navy.

Acknowledgements

First, I would offer my gratitude to my advisor, Dr. Stephen L. Campbell. In addition to being a brilliant and successful scholar, he is also a patient and generous mentor.

I am grateful to the members of my committee: Dr. Pierre Gremaud, Dr. Ralph Smith, and Dr. Hien Tran. I am grateful to them for ensuring the quality of my work as well as for being excellent teachers. I would offer a special thanks to Dr. Gremaud who so cheerfully gave me many hours of his time through four courses and the qualifying exams, preparing me to do this work.

I would also like to thank Dr. John Betts and The Boeing Company, as well as Dr. Ramine Nikoukhah and INRIA-Rocquencourt. My time working with these men at their respective institutions was invaluable to my education and a true pleasure.

I also offer my thanks to the National Science Foundation, which supported this work.

I would like to thank Dr. Leona Harris-Clark and Dr. Doug Cochran. The support of close friends is invaluable.

I am grateful to my family for their patience and understanding.

Finally, I would like to thank God. Without Him, none of this would have been possible.

Table of Contents

List of Figures

Chapter 1

Introduction and Background

Fault, or failure, detection is a process by which it is determined whether a system is working in a manner which is 'normal'. Examples include determining that the pump in a sewage plant is malfunctioning, the brakes on a car are performing inadequately, or the engine on an aircraft is experiencing problems. In each of these cases, determining a system failure allows for a controlled shut down of the system, preventing a catastrophic event. Beyond simple detection is model identification or isolation. In the case of model identification, there are multiple models, each representing a particular 'failure' of the system. Determining which model represents the system at the time of the test then determines the specific nature of the system failure.

Recently an approach for robust failure detection and multi-model identification in the presence of bounded energy noise over short time intervals has been introduced [12] . The fault detection and model identification (FDMI) algorithm sets up and solves an optimization problem, the solution of which is a detection, or auxiliary, signal. The detection signal is designed so that the model output sets are separated when it is applied to the system. Once the signal is applied, the system output is tested in a hyperplane test. The hyperplane test is done in real time but the computation of the detection signal is done offline.

In this thesis, we extend the approach to systems which include delays [11, 8, 10], in addition to looking at whether it is possible to make the model identification decision early [9]. In Chapter 1, we discuss background and prior research. More specifically, Section 1.1 reviews mathematical background related to the research. Section 1.2 offers a broader view of fault detection and model identification, providing context for the research in this thesis. Then in Section 1.3, we review the particular approach to fault detection and model identification that we extend.

1.1 Mathematical Background

In this section, we briefly review some mathematical concepts which are relevant to this thesis. There is a huge body of work associated with each topic and they are the subject of many books. Rather than trying to even broadly review them, which would be a book in its own right, we simply include the barest essentials which will be used throughout this thesis.

1.1.1 Optimal Control

The FDMI algorithm solves the fault detection problem by finding and utilizing a detection signal. The detection signal is the solution to an optimal control problem. In addition, the FDMI algorithm will use the necessary conditions of another optimal control problem as constraints when finding the detection signal. Because of the fundamental role it plays in this work, we briefly discuss the optimal control problem and the necessary conditions needed to solve it.

A subproblem of the auxiliary signal design algorithm is finding the necessary conditions of the following optimal control problem:

$$\begin{aligned} \min_{x_0,\mu} \quad J &= \phi(x_0) + \int_{t_0}^{t_f} L(t,x,\mu)\, dt && \text{(1.1a)} \\ \text{subject to} \quad \dot{x} &= f(t,x,\mu) && \text{(1.1b)} \\ 0 &= g(t,x,\mu) && \text{(1.1c)} \end{aligned}$$

where t_0 and t_f are fixed. x is a state variable and μ is a control. Both are dependent on the time, t. J is called the cost or performance index of our problem. Using the control, μ, our objective is to minimize J while constrained by the dynamics (1.1b) and algebraic constraint (1.1c).

$L(t,x,\mu)$ may have many physical interpretations. It may represent energy with the goal of the problem being to minimize the energy in a system. The function ϕ represents some weighted function of the initial condition. Note that optimal control problems are often formulated as functions of the final state, rather than the initial state. In this, our problem diverges from what might typically be found in a text book. There are many other types of optimal control problems, with different costs and different constraints; but, we focus on this problem here as it is the problem related to our algorithms.

The constrained minimum of J is at a stationary point of J', or $J' = 0$, where J' is formed by adjoining the constraints to the cost with multipliers. That is, we set up

$$J' = \phi(x_0) + \int_{t_0}^{t_f} [L + \lambda^T(f - \dot{x}) + \eta^T g] \;\; dt.$$

Letting $H(t, x, \mu) = L + \lambda^T f + \eta^T g$ be the Hamiltonian, we have

$$J' = \phi(x_0) + \int_{t_0}^{t_f} [H - \lambda^T \dot{x}] \; dt.$$

To find the unconstrained minimum of J', we determine when $dJ' = 0$. Using Leibnitz rule[1] and incrementing J' as a function of increments in x, λ, η, and μ, we have

$$dJ' \;\; = \;\; \phi_x^T \; dx(t_0) + \int_{t_0}^{t_f} [H_x^T \; \delta x + H_\mu^T \; \delta\mu - \lambda \; \delta\dot{x} + (H_\lambda - \dot{x})^T \; \delta\lambda + H_\eta \; \delta\eta] \;\; dt.$$

Note that since our problems have fixed initial and final times, any variations in these are 0 and are not included in dJ'. Eliminating the variation in $\dot{x}$ with an integration by parts and using the fact that $dx(t_0) = \delta x(t_0)$ and $dx(t_f) = \delta x(t_f)$ since there is no variation in initial or final time, we have

$$\begin{aligned} dJ' \;\; &= \;\; (\phi_x^T + \lambda(t_0)) \; dx(t_0) + (-\lambda(t_f)) \; dx(t_f) \\ &+ \int_{t_0}^{t_f} [(H_x^T + \dot{\lambda}^T) \; \delta x + H_\mu^T \; \delta\mu + (H_\lambda - \dot{x})^T \; \delta\lambda + H_\eta \; \delta\eta] \;\; dt. \end{aligned}$$

Then to determine when $dJ' = 0$, we set the coefficients of $\delta\lambda, \delta x, \delta\eta, \delta\mu, dx(t_0)$ and $dx(t_f)$

[1] *Leibnitz rule*: if $x(t) \in R^n$ is a function of t and $\int_{t_0}^{t_f} h(x(t), t) \; dt$ where $J(\cdot)$ and $h(\cdot)$ are both real scalar functionals, then $dJ = h(x(t_f)t_f)dt_f - h(x(t_0), t_0)dt_0 + \int_{t_0}^{t_f} [h_x^T(x(t), t)\delta x] \; dt$. [44]

to 0. We arrive at the conditions

$$\begin{aligned}
\dot{x} &= H_\lambda = f(t,x,\mu) \\
-\dot{\lambda} &= H_x = L_x + \lambda^T f_x + \eta^T g_x \\
0 &= H_\eta = g \\
0 &= H_\mu = L_\mu + \lambda^T f_\mu + \eta^T g_\mu \\
\lambda(t_0) &= -\phi_x \\
\lambda(t_f) &= 0.
\end{aligned}$$

More information on optimal control theory can be found in [44, 2, 45, 28].

1.1.2 Solving the Optimal Control Problem

In order to solve our detection problem, we will need to use optimization software. We use the package Sparse Optimal Control Software (SOCS) [7, 6] provided by The Boeing Company. SOCS is a package of FORTRAN subroutines which can solve complex optimization problems.

The problem that SOCS has to solve includes not only the necessary conditions for (1.1) but also inequality constraints and has the more general form

$$\min_{t_0 \le t \le t_f} \quad J(t,x,u)$$

subject to

$$\begin{aligned}
\dot{x} &= f(t,x,u) && (1.7a) \\
0 &= g(t,x,u) && (1.7b) \\
c_1 &\le h(t,x,u) \le c_2 && (1.7c)
\end{aligned}$$

where t_0 and t_f are fixed. As a direct transcription code, SOCS first discretizes the problem across the entire time interval. It divides the time interval $[t_0\ t_f]$ into n parts by selecting mesh points $t_0 < t_1 < \ldots t_{n-1} < t_n = t_f$. Then a discretization scheme is applied, forming a nonlinear programming problem. Several discretization methods are available to the user of SOCS. Below we will briefly review a few of them.

Let $u_k = u(t_k)$. Let $\tilde{t}_k = \frac{1}{2}(t_k + t_{k+1})$ and let $\tilde{u}(\tilde{t}_k) = \tilde{u}_k$. Let $h_k = t_k - t_{k-1}$.

1. Euler's Method is a simple first order method. The discretization is given by

$$x_k = x_{k-1} + h_k(f_k)$$

   where $f_k = f(t_k, x_k, u_k)$.

2. Compressed Hermite-Simpson is a fourth order method. The discretization is given by

$$x_k = x_{k-1} + \frac{h_k}{6}(f_k + 4\tilde{f}_k + f_{k-1})$$

   where $\tilde{f}_k = f(\tilde{t}, \tilde{x}_k, \tilde{u}_k)$ and $\tilde{x}_k = \frac{1}{2}(x_{k-1} + x_k) + \frac{h_k}{8}(f_{k-1} + f_k)$.

3. The trapezoidal rule is a second order method. The discretization is given by

$$x_k = x_{k-1} + \frac{h_k}{2}(f_{k-1} + f_k)$$

   where $f_k = f(t_k, x_k, u_k)$.

4. The classic 4-stage Runge-Kutta method is a fourth order method. The discretization is given by

$$x_k = x_{k-1} + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

   where

$$\begin{aligned} k_1 &= h_k f(t_{k-1}, x_{k-1}, u_{k-1}) \\ k_2 &= h_k f(\tilde{t}_k, x_{k-1} + \frac{k_1}{2}, \tilde{u}_k) \\ k_3 &= h_k f(\tilde{t}_k, x_{k-1} + \frac{k_2}{2}, \tilde{u}_k) \\ k_4 &= h_k f(t_k, x_{k-1} + k_3, u_k). \end{aligned}$$

For more information on discretization and numerical integration, see [13, 59, 55, 27].

Once the problem has been discretized, then SOCS solves it using a Sequential Quadratic Programming (SQP) technique. An SQP algorithm solves the nonlinear programming problem by solving a sequence of smaller problems [25]. For each subproblem, it replaces the objective function with the quadratic approximation and replaces the constraint functions by linear approximations.

Once SOCS finds a solution to the NLP problem, it assesses the accuracy of the solution. Then if necessary, it refines the mesh that it is working on and starts the optimization again with a finer discretization.

For more information on optimization, see [7, 40, 45, 20, 2, 5].

1.1.3 Approximation Theory

Approximation theory is the study of how quantities can be approximated by other quantities, usually by simpler quantities. It is also concerned with the error in the approximation process. In this section we will briefly review some of the basic ideas of approximation theory.

A complicated function $f(t)$ is approximated by a less complicated function $\phi(t,a)$ where $a = [a_0, a_1, \ldots, a_n]$ are parameters characterizing the approximation. There are 3 basic types of approximation approaches, which are based on how one defines a 'best' approximation [18].

1. *Interpolation*: In the case of interpolation, the parameters, a_i, are chosen so that at a fixed set of points $\{t_i\}$ for $i = 0, 1, \ldots, n$, the interpolating function, ϕ, and the true function,f, agree both in function value and sometimes at one or more derivatives. That is, $\phi(t_i, a) - f(t_i)$ for $i = 0, 1, \ldots, n$.

2. *Least Squares*: In a least squares approximation, the parameters are chosen so as to minimize the expression $\|f(t) - \phi(t,a)\|_2$ where $\|\cdot\|_2$ is the 2-norm.

3. *Min-max*: In a min-max approximation, the parameters are chosen so as to minimize the expression $\|f(t) - \phi(t,a)\|_\infty$ where $\|\cdot\|_\infty$ is the max-norm.

Spline Interpolation

Of particular interest to us will be a local approximation method known as spline interpolation. With spline interpolation, we approximate our function on subintervals with polynomials of small degree. In Section 5.1 , we will discuss a linear, piecewise polynomials in a spline method at length. Also common are the cubic splines. In this case, the approximating functions are piecewise, cubic polynomials. They are chosen to agree with the true function values at the end points of the subintervals. In addition, the first and second derivatives are made to be continuous between intervals [18].

For more information on approximation theory, in general, and splines, in particular, see [36, 34, 18, 59, 45].

1.2 FDMI: Introduction and Prior Research

Introduction

In an increasingly automated society, fault, or failure, detection plays a more vital role in system maintenance and stability all the time. For a myriad of reasons, understanding how a system is functioning while it is operating is of much value. Obvious examples include understanding how the pump in a sewage plant is working or understanding how the reactor in a nuclear power plant is functioning. Maintenance of these systems is crucial to public health, safety and comfort. Allowing a breakdown of these systems could be an environmental disaster. Yet, there is a desire to maintain the systems in a cost effective manner with minimal disturbance to the system as it is running.

Related to fault detection is model identification. In this case, the system is modelled mathematically by two or more models, each representing some particular state of the system. Model identification, or isolation, means determining which model best represents the current state of the system. One of the models represents the properly working system, while the other models represent failure states. Using different models to represent individual failures allows for very specific detection.

Passive and Active Methods

There are two basic approaches to the fault detection problem: active and passive. In a passive approach, system inputs and outputs are monitored without direct interaction with the system. The detector must determine if a failure has occurred only from monitoring inputs and outputs and executing some sort of comparison to ‘normal’. An unfortunate problem associated with passive failure detection is that failures can be hidden and go undetected. An example of this limitation, taken from Campbell and Nikoukhah [12], is the following,

> A more dramatic example was in 1987 when a pilot flying an F-117 Nighthawk, which is a twin tailed aircraft known as the stealth fighter, encountered bad weather during a training mission. He lost one of his tail assemblies and proceeded back and landed his plane without ever knowing that he was missing part of the tail. The robustness of the control system in this case had the beneficial effect of enabling the pilot to return safely. However, it also had the effect that the pilot did not realize that his aircraft had reduced capability and that the plane would not have performed correctly if a high speed maneuver was required.

In contrast to the passive approach is the active approach. Direct interaction with the system is possible. Rather than continuously monitoring the system, as is usually the case with a passive approach, in the active approach, an input known as a detection signal or auxiliary signal is periodically put in the system for a short test period. During the detection period, system outputs are monitored and used to answer the detection question. An example of an active fault detection test might be tapping the brakes on a car in the rain. The driver taps the brakes to ensure that they are working, before they are needed to prevent a collision. The tap is an input to the system, allowing the driver to examine the output or reaction of the car.

Campbell, Nikoukhah, et al. have written several papers on the subject of active fault detection and model identification. This thesis is an extension of a method they developed with K. Horton which utilizes an auxiliary signal. The auxiliary signal is designed so that with the application of a hyperplane test to the output of the system, the detection decision can be made. The reader is referred to [51, 50, 15, 14, 52]. More details on the method will be given in Section 1.3.

1.2.1 Other Prior Research

According to Frank [21], there are essentially two ways of generating signals for use in model-based detection: state estimators and parameter estimators. Much work has been done in both of these areas in the last twenty years or so. There are several good survey articles [21, 32, 23] and books [12, 4] available on the topic of fault detection and model identification. Below we review prior research on fault detection and identification. The review is broken into two parts. The first part is a more general review of detection and identification on delayed systems. The second part is specifically about approximation of delayed systems, as approximation methods are a pivotal part of our research.

Detection, Identification and Delays

In [62], Zhang et al. look at a detection on nonlinear, delayed systems where they attempt to identify unknown inputs. Using a feedback control which is supposed to act as the 'model' of the system, the state of the system is compared to that of the plant. From this, they use a sequence of feedback controls to construct and approximate the unknown control. In [29], Hartfung et al. solves a parameter estimation problem using an Euler-type approximation. Hu and Gu [26] propose two algorithms using piecewise Chebyshev polynomials to examine a parameter estimation problem on linear time-varying delay systems. According to the authors, these algorithms are superior to those derived from shifted Chebyshev polynomials.

In [56], Pourboghrat and Chyung look at parameter identification on a class of linear delay equations. The authors use a recursive least squares identification algorithm for online identification of system matrices. Banks, et al. [3] look at system identification and parameter estimation, as well as estimation of delay. In addition to introducing applications, they develop two approaches. The first approach involves an averaging approximation and the second is a spline approximation.

Another important class of problems is stochastic models with delays. Although we will not go into these systems beyond these remarks, we refer the interested reader to [19, 17, 64, 63, 61, 43, 53] for some articles on these systems.

Approximation of Delayed Systems

As we will see, approximation of delayed systems will be an important part of this thesis. Thus, we take a few minutes to specifically review some of the work done in this field. We will use an approximation schemed developed by Ito and Kappel using linear, piecewise splines found in *A Uniformly Differentiable Approximation Scheme for Delay Systems Using Splines* [34]. Ito and Kappel have done much important work in this field and other works of theirs include some thoughts on approximation of infinite delay and Volterra type equations [33], approximation of Cauchy problems [35], and approximation of PDE's [36]. In addition, they wrote *Evolution Equations and Approximations* [37].

Kappel has also been author or co-author of a number of other papers, including a survey on the approximation of Linear Quadratic Regulator (LQR) problems for delay systems [39] and a spline approximation for autonomous nonlinear functional-differential equations[38].

In [57], Prager discusses a one parameter family of spline-type approximation schemes of the system $\dot{x}(t) = A_0x(t) + A_1x(t-r) + f(t)$ based on the transformation of the system into the abstract Cauchy problem $\dot{z}(t) = Az(t) + (f(t), 0)$, with the state space $M^2 = \mathbf{R}^2 \times L^2(-r, 0; \mathbf{R}^n)$, $z(0) = \phi$, and $\phi \in M^2$.

Partington and Mäkilä write about a shift operator induced approximation of delay systems in [47]. The method allows one to write low-order approximations of the delay system. On a similar theme, they write about Laguerre and Kautz shift approximations of delay systems in [46] and Partington also writes about approximation of certain delay systems by Fourier-Laguerre series in [54].

In [42], Lee and Tsay approximate solutions for linear time-delay systems with first order Pad*é* approximation and orthogonal polynomial expansions. Glader et al. propose a Hankel optimal rational approximation for delay systems in [24] and compare it to Pad*é* approximation of delay systems and to the Carathodory-Fej*é*r method for real rational approximation of Trefethen and Gutknecht as found in [60].

1.3 Fault Detection via the Minimum Energy Detection Signal

In this section, we describe the minimum energy detection signal algorithm as developed in [30]. This approach to fault detection and model identification has two steps. The first step is finding an auxiliary signal which is the result of a carefully designed, non-standard optimal control problem. The signal is designed so that when it is applied to the system, the output sets from each model will be disjoint. Finding the detection signal can be computationally intensive but can be done offline.

The second step of solving the fault detection problem is the design of a hyperplane test. After application of the test signal to the system, the convex output sets are disjoint. As that they are convex and disjoint, we know there exists a hyperplane between them. Using a test designed around this hyperplane, the system output can be used to determine from which model the output originated, completing the model identification.

What follows is an intuitive description of the signal detection algorithm and the hyperplane test, intended to serve as a basis and background for the rest of this thesis. For a more rigorous re-examination, see [30, 12, 52, 14, 15, 51].

1.3.1 Minimum Energy Detection Signal

Problem Set Up

Assume we have a system which can be modelled by one of the following models

$$\dot{x}_i = A_i x_i + B_i v + M_i \mu_i \tag{1.9a}$$

$$y_i = C_i x_i + N_i \mu_i \tag{1.9b}$$

for $i = 0$ and 1, for $t \geq 0$. Here x_i, y_i, v, μ_i are the system states, output, detection signal and noise, respectively. A_i, B_i, C_i, M_i, N_i are matrices of appropriate dimension. v and μ_i are taken to be in $L^2[0,\omega] = L^2$. This implies that x_i and y_i are in L^2, as well. We can say that model 0 represents the properly working system, while the rest of the models each represent a failure of the system. Note that the only commonality between the two models is the detection signal, v. For clarity in exposition, in this thesis we only discuss two constant, linear coefficient models. However, the approach, with appropriate modifications, is applicable to more than two models, as well as linear, time-varying models. In the case that there are more than two models, an appropriate test signal can be found to separate all of the output sets. The hyperplane test can only be applied to two models at a time and used to determine which model the output did *not* come from. However, by comparing two models at a time, eliminating the possibilities, the model identification decision can be made.

Without some kind of bound on the noise, any output would be possible from either model and perfect model identification would be impossible. Thus, we assume that the noise is bounded in the L^2 norm. That is, we let S_i be the noise measure where

$$\mathcal{S}_i(x_i(0), \mu_i) = x_i(0)^T Q_i x_i(0) + \int_0^\omega |\mu_i(t)|^2 \quad dt < 1, \quad i = 0, 1. \tag{1.10}$$

Note that the choice of 1 as the bound is for simplicity. We could have easily chosen a different bound (see Lemma 2.2 of [30]). The bound of (1.10) differs from much of the failure detection literature where stochastic assumptions are made on the noise, μ_i. The approach based on (1.10) allows for a variety of problems including some with small nonlinearities and disturbances with unknown statistical properties. There are alternatives discussed in [12]. Once the noise is bounded, we can consider the outputs of the models.

Let $\mathcal{A}_i(v)$ be the output set of model i for $i = 0$ and 1, where model 0 represents the properly working system and model 1 represents the faulted or failing system. Perfect fault detection would imply that

$$\mathcal{A}_0(v) \cap \mathcal{A}_1(v) = \emptyset. \tag{1.11}$$

Define $\mathcal{L}_i(f) = \int_0^T e^{A_i(t-s)} f(s)ds$ so that $\mathcal{L}_i(f)$ is the solution of $\dot{z} = A_i z + f$ with $z(0) = 0$. Let ξ_i be the free initial condition for ODE (1.9). Then

$$y_i = \bar{y}_i + (C_i \mathcal{L}_i M_i + N_i)\mu_i + C_i e^{A_i} \xi_i$$

where $\bar{y}_i = C_i \mathcal{L}_i(B_i v)$ is a vector which is linearly dependent on the detection signal, $\{(C_i \mathcal{L}_i M_i + N_i)\mu_i : \|\mu_i\| < 1\}$ is an open, convex set, and $\{C_i e^{A_i t} \xi_i : \xi_i \in \mathbf{R}^N\}$ is a finite dimensional subset of $L^2[0, \omega]$.

Thus, the output sets, $\mathcal{A}_i(v)$, are translates of open sets by $\bar{y}_i$ and they are affinely dependent upon the detection signal, v. We say a detection signal is *proper* if it gives disjoint output sets. That is, the detection signal v, is proper if the translation by the vector $\bar{y}_i$ pushes the sets apart. As we are looking for the *minimum energy* detection signal, then we will find a proper detection signal of minimum norm. Figure 1.3.1 illustrates a proper detection signal pushing the output sets apart. A detection signal is *strictly proper* if the output sets are a positive distance apart.

Formulating the problem in terms of the definition of *proper*

A proper detection signal separates the output sets of the two models. We now consider the definition of proper in detail and develop a computational criteria for a proper detection signal. If (x_0, μ_0, x_1, μ_1) satisfies the models (1.9) *and* satisfies the noise bound (1.10), then a proper detection signal, v, ensures that the output sets of the models will be distinct. Another way to look at this is to say, if (x_0, μ_0, x_1, μ_1) satisfy the models *and* v is a proper detection signal, *but* the outputs are still *not distinct*, then (x_0, μ_0, x_1, μ_1) must not satisfy

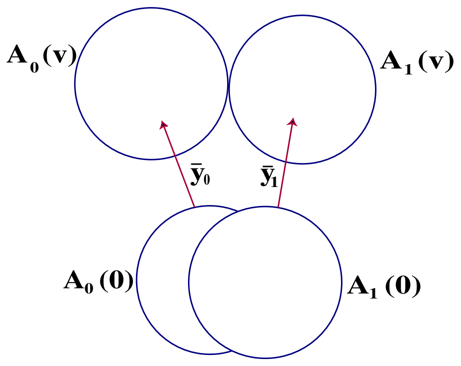

Figure 1.1: The figure illustrates a proper detection signal, v, pushing the output sets, $\mathcal{A}_i(v)$, apart.

the noise bound. Recall that we assumed that the noise in the models was bounded. That is, we assumed

$$\mathcal{S}_i(x_i(0), \mu_i) = x_i(0)^T Q_i x_i(0) + \int_0^\omega |\mu_i(t)|^2 \; dt < 1, \quad i = 0, 1. \tag{1.12}$$

Thus, if (x_0, μ_0, x_1, μ_1) satisfies the models *and* v is a proper detection signal, *but* an output, y, is still in both output sets, it must be that

$$\max\{\mathcal{S}_0(x_0(0), \mu_0), \mathcal{S}_1(x_1(0), \mu_1)\} \geq 1. \tag{1.13}$$

As this must be true *for all* possible (x_0, μ_0, x_1, μ_1) which satisfy the models but do not have distinct outputs, it is sufficient to ensure it is true for the minimum of them. Thus, we have

$$\min_{x_i, \mu_i} \max\{\mathcal{S}_0(x_0(0), \mu_0), \mathcal{S}_1(x_1(0), \mu_1)\} \geq 1.$$

With this in mind, we can give a technical characterization of the term *proper*.

Lemma 1 (Proper Detection Signal). *The auxiliary signal, v, is* proper *if and only if for all $(x_0, \mu_0, x_1, \mu_1, y)$ satisfying*

$$\begin{aligned} \dot{x}_1 &= A_0 x_0 + B_0 v + M_0 \mu_0 \\ y &= C_0 x_0 + N_0 \mu_0 \end{aligned}$$

and

$$\begin{aligned}\dot{x}_1 &= A_1x_1 + B_1v + M_1\mu_1\\ y &= C_1x_1 + N_1\mu_1\end{aligned}$$

then

$$\min_{x_i,\mu_i} \max\{\mathcal{S}_0(x_0(0),\mu_0), \mathcal{S}_1(x_1(0),\mu_1)\} \geq 1. \tag{1.16}$$

v is called not proper *if (1.16) does not hold.*

The inner maximization is not continuous in the sense that it is over the discrete set i. Thus, the next step in formulating the optimal control problem which we will solve to find the test signal is to replace the discontinuous maximum problem with a continuous one and to switch the max and min (see Theorem 2.1 in [30]). That is, we replace

$$\min_{x_i,\mu_i} \max\{\mathcal{S}_0(x_0(0),\mu_0), \mathcal{S}_1(x_1(0),\mu_1)\} \geq 1$$

with

$$\max_{0\leq\beta\leq 1} \{\min_{x_i,\mu_i} \ \beta\mathcal{S}_0(x_0(0),\mu_0) + (1-\beta)\mathcal{S}_1(x_1(0),\mu_1)\} \geq 1$$

Then finding the minimum energy, proper detection signal is equivalent to finding

$$\min \|v\|^2$$

such that

$$\max_{0\leq\beta\leq 1} \min_{x_i,\mu_i} \beta\mathcal{S}_0(x_0(0),\mu_0) + (1-\beta)\mathcal{S}_1(x_1(0),\mu_1) \geq 1$$

subject to

$$\begin{aligned}\dot{x}_i &= A_ix_i + B_iv + M_i\mu_i\\ y &= C_ix_i + N_i\mu_i\end{aligned}$$

for $i = 0, 1$.

We note that the output y in both models is the same. Thus, we can eliminate the output by replacing

$$y = C_ix_i + N_i\mu_i$$

for $i = 0, 1$ with

$$0 = C_0 x_0 - C_1 x_1 + N_0 \mu_0 - N_1 \mu_1.$$

If we let

$$\dot{x} = Ax + Bv + M\mu \quad \text{be} \quad \begin{cases} \dot{x}_0 = A_0 x_0 + B_0 v + M_0 \mu_0 \\ \dot{x}_1 = A_1 x_1 + B_1 v + M_1 \mu_1 \end{cases}$$

and let

$$0 = Cx + N\mu \qquad \text{be} \qquad 0 = C_0 x_0 - C_1 x_1 + N_0 \mu_0 - N_1 \mu_1$$

then we can let

$$J_v(\beta) = \begin{cases} \min_{x_i,\mu_i} \beta \mathcal{S}_0(x_0(0), \mu_0) + (1-\beta)\mathcal{S}_1(x_1(0), \mu_1), \\ \text{subject to} \\ \dot{x} = Ax + Bv + M\mu \\ 0 = Cx + N\mu. \end{cases}$$

Using these terms, the problem of finding the minimum energy detection signal becomes:

$$\text{Find} \qquad \min \|v\|^2$$

such that

$$\max_{0 \le \beta \le 1} J_v(\beta) \ge 1.$$

The final step in the process of formulating the optimal control problem we will solve to find the minimum energy signal is to find the necessary conditions of the $J_v(\beta)$ problem. That is, we need to find

$$\min_{x_i,\mu_i} \beta \mathcal{S}_0(x_0(0), \mu_0) + (1-\beta)\mathcal{S}_1(x_1(0), \mu_1) \tag{1.18}$$

subject to

$$\begin{aligned} \dot{x} &= Ax + Bv + M\mu \\ 0 &= Cx + N\mu. \end{aligned}$$

Forming the Hamiltonian of this system, we get

$$H = \frac{1}{2} x_0^T Q_\beta x_0 + \frac{1}{2} \mu^T V_\beta \mu + \lambda(-\dot{x} + Ax + Bv + M\mu) + \eta(Cx + N\mu)$$

where

$$\frac{1}{2}x_0^T Q_\beta x_0 = \beta x_0(0)^T Q_0 x_0(0) + (1-\beta)x_1(0)^T Q_1 x_1(0) \tag{1.20}$$

and

$$\frac{1}{2}\mu^T V_\beta \mu = \beta\|\mu_0\|^2 + (1-\beta)\|\mu_1\|^2. \tag{1.21}$$

Using the Euler equations, we get the necessary conditions

$$\begin{aligned} \dot{x} &= Ax + Bv + M\mu \\ \dot{\lambda} &= -A^T\lambda + C^T\eta \\ 0 &= V_\beta\mu - M^T\lambda + N^T\eta \\ 0 &= Cx + N\mu \\ \lambda(0) &= -Q_\beta x(0), \quad \lambda(\omega) = 0. \end{aligned}$$

Then the optimization problem for finding the minimum energy, proper detection signal is to minimize the detection signal subject to

1. An expression for the value of J_β.
2. The necessary conditions for the J_β problem.
3. The constraint $J_\beta \geq 1$.

This is equivalent to finding

$$\min \|v\|^2$$

$$\begin{aligned} \text{subject to} \quad \dot{x} &= Ax + Bv + M\mu \\ \dot{\lambda} &= -A^T\lambda + C^T\eta. \\ \dot{Z} &= \frac{1}{2}\mu^T V_\beta \mu \\ 0 &= V_\beta\mu - M^T\lambda + N^T\eta \\ 0 &= Cx + N\mu \\ Z(0) &= \frac{1}{2}x_0^T Q_\beta x_0, \quad Z(\omega) \geq 1 \\ \lambda(0) &= -Q_\beta x(0), \quad \lambda(\omega) = 0 \\ & 0 < \beta < 1. \end{aligned}$$

The minimum energy detection signal algorithm (MEDS)

Having developed the optimization problem for solving the minimum energy detection signal problem, we can now describe the algorithms and precise problems which are solved. Included with each step of the algorithm is the name of the software which can be used in that step. As we used MATLAB, by The MathWorks, Inc. [31], and SOCS, by The Boeing Company [7], they are given. However, other software packages would work just as well. In addition to MATLAB and SOCS, Maple, by Waterloo Maple, Inc. [49], was used to generate the FORTRAN subroutines needed for SOCS. Note that the description below includes a change in variable resulting in a reduction of the dimensions of the problem and rewriting the $J_v(\beta)$ problem in LQR form. As it added little to the understanding of the approach, these things weren't mentioned above. However, they are included here for completeness. In [30], Horton includes a Riccati form of the problem but that is also not included here as we have not used it.

The minimum energy detection signal(MEDS) algorithm:

1. Perform QR decomposition on N_i^T (MATLAB)

   $$N_i^T = Q_i R_i$$

   where the Q_i are unitary. Then

   $$N_i = R_i^T Q_i^T.$$

2. Perform constant orthogonal coordinate changes on μ_i (MATLAB).

   (a) Let $R_i^T = \begin{bmatrix} \overline{N}_i & 0 \end{bmatrix}$ where $\overline{N}_i$ is invertible.

   (b) Let $Q_i^T \mu_i = \begin{bmatrix} \overline{\mu}_i \\ \widetilde{\mu}_i \end{bmatrix}$ with the same partitioning as R_i^T. Thus,

   $$N_i \mu_i = \begin{bmatrix} \overline{N}_i & 0 \end{bmatrix} \begin{bmatrix} \overline{\mu}_i \\ \widetilde{\mu}_i \end{bmatrix}.$$

(c) Let $M_iQ_i^{-T} = \left[\overline{M}_i\widetilde{M}_i\right]$ with the same partitioning as R_i^T. Thus

$$M_i\mu_i = M_iQ_i^{-t}Q_i^T\mu_i = \left[\overline{M}_i\widetilde{M}_i\right]\begin{bmatrix}\overline{\mu}_i\\ \widetilde{\mu}_i\end{bmatrix}.$$

Now, the system model is

$$\dot{x} = A_i + B_i + \overline{M}_i\overline{\mu}_i + \widetilde{M}_i\widetilde{\mu}_i \tag{1.25a}$$

$$y = C_ix_i + \overline{N}_i\overline{\mu}_i. \tag{1.25b}$$

3. Reduce model dimension by eliminating y and $\overline{\mu}_0$ (MATLAB).

(a) Combine both equations ($i = 0, 1$) for y, solve for $\overline{\mu}_0$, and substitute into (1.25) for $i = 0$. Then,

$$\dot{x}_0 = (A_0 - \overline{M}_0\overline{N}_0^{-1}C_0)x_0 + \overline{M}_0\overline{N}_0^{-1}C_1x_1 + B_0v + \widetilde{M}_0\widetilde{\mu}_0 + \overline{M}_0\overline{N}_0^{-1}\overline{N}_1\overline{\mu}_1.$$

(b) Let

$$x = \begin{bmatrix}x_0\\ x_1\end{bmatrix}, A = \begin{bmatrix}A_0 - M_0N_0^{-1}C_0 & M_0N_0^{-1}C_1\\ 0 & A_1\end{bmatrix}, \ B = \begin{bmatrix}B_0\\ B_1\end{bmatrix},$$

$$M = \begin{bmatrix}\widetilde{M}_0 & \overline{M}_0\overline{N}_0^{-1}\overline{N}_1 & 0\\ 0 & \overline{M}_1 & \widetilde{M}_1\end{bmatrix}, \ \mu = \begin{bmatrix}\overline{\mu}_0\\ \overline{\mu}_1\\ \widetilde{\mu}_i\end{bmatrix}.$$

The system model is now

$$\dot{x} = Ax + Bv + M\mu.$$

4. Compute new system matrices (MATLAB).

(a) Let $C = \begin{bmatrix}C_0 & -C_1\end{bmatrix}$ and $N = \begin{bmatrix}0 & \overline{N}_1 & 0\end{bmatrix}$, with the columns of N conforming to μ.

(b) Let $H = -4\beta C^TN$ and $Q = 2\beta C^TC$.

(c) Let

$$R = 2\begin{bmatrix} \beta I & 0 & 0 \\ 0 & (1-\beta)I + \beta\overline{N}_1^T\overline{N}_1 & 0 \\ 0 & 0 & (1-\beta)I \end{bmatrix}$$

with the rows and columns of R conforming to μ.

(d) Compute A, B, M, H, Q, R.

5. Perform one of the following constrained optimization problems (SOCS)

(a) Problem formulation 1:

$$(\gamma^*)^2 = \max_{v,\beta} Z(\omega) \tag{1.30}$$

subject to the constraints

$$\begin{aligned} \dot{x} &= Ax + Bv + M\mu && (1.31a) \\ \dot{\lambda} &= -Qx - \frac{1}{2}H\mu - A^T\lambda,\ \lambda 0 = \lambda\omega = 0 && (1.31b) \\ \dot{\theta} &= v^Tv,\ \theta(0) = 0,\ \theta(\omega) = 1 && (1.31c) \\ \dot{Z} &= \frac{1}{2}[x^TQx + x^TH\mu + \mu^TR\mu],\ Z(0) = 0 && (1.31d) \\ 0 &= R\mu + \frac{1}{2}H^Tx + M^T\lambda && (1.31e) \\ & 0.01 < \beta < 0.99. && (1.31f) \end{aligned}$$

(b) Problem formulation 2:

$$(\gamma^*)^2 = \min_{v,\beta} \int_0^\omega \parallel v \parallel^2 dt$$

subject to the constraints

$$\begin{aligned} \dot{x} &= Ax + Bv + M\mu \\ \dot{\lambda} &= -Qx - \frac{1}{2}H\mu - A^T\lambda,\ \lambda(0) = \lambda(\omega) = 0 \\ \dot{Z} &= \frac{1}{2}[x^TQx + x^TH\mu + \mu^TR\mu],\ Z(0) = 0,\ Z(\omega) \geq 1 \\ 0 &= R\mu + \frac{1}{2}H^Tx + M^T\lambda \\ & 0.01 < \beta < 0.99. \end{aligned}$$

The constraint (1.31f) uses .01 and .99 rather than 0 and 1 to bound β away from 0 and 1 early in the optimization process.

1.3.2 Model Identification by the Separating Hyperplane

Applying the detection signal from the MEDS algorithm guarantees that the output sets from the models are distinct, convex sets. However, it does not reveal the model with which the output is associated. In order to determine from which model the output originated, other steps need to be taken. As we know that application of the detection signal causes the output sets to be disjoint and convex, we know that there exists a hyperplane separating them. Using this hyperplane, we can devise a test function, ϕ, and the identification can be made.

The Problem

Let

$$\dot{x}_i = A_i x_i + B_i \hat{v} + M_i \mu_i \tag{1.34a}$$

$$y_i = C_i x_i + N_i \mu_i \tag{1.34b}$$

for $i = 0$ and 1, be the previously defined normal and failed models with the exception that $\hat{v}$ is now the *minimum energy detection signal* as found in the MEDS algorithm. Also, as we know that the minimum energy signal provides distinct output sets, y now has an index.

Due to the design of the detection signal, we know that the output sets, $\mathcal{A}_i(v)$, are open, convex sets. We also know that while the output sets are distinct since we found a proper, minimum energy detection signal, the closures of the output sets are not distinct and must share at least one point. The reason for this is that the detection signal is minimal. If a detection signal which is infinitesimally smaller than the minimum energy detection signal were applied then the signal would not be proper and the output sets would not be disjoint. However, if the open output sets intersect with the application of an infinitesimally smaller detection signal, the closures of the output sets must intersect with the application of the minimum energy, proper detection signal.

It is possible that the closures might have more than one intersection. This could happen when both of the sets are not strictly convex. Should the A_i matrices share a common eigenvector, then the sets have a straight, parallel side. A matrix manipulation can eliminate this problem and it is assumed that this problem has been taken care of if it should occur.

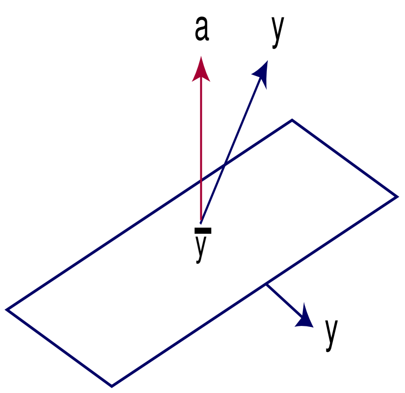

Figure 1.2: Using the hyperplane, a test can be formulated that will determine if y lies above the plane or below it, determining from which model the output is produced.

Let us assume there is a unique point of intersection. This is guaranteed if at least one of the sets is strictly convex. Also, recall that the equations $y = C_i x_i + N_i \mu_i$ for $i = 0, 1$ were combined to eliminate the y. Thus, when the optimal trajectory x_1 and the optimal noise μ_1 are substituted into this expression, the resulting y is the point of intersection of the closures of $\mathcal{A}_i(v)$. Let the resulting y be $\overline{y}$.

Because the output sets are convex and disjoint, there exists a hyperplane passing through $\overline{y}$ which separates the sets. Using this hyperplane, a test can be formulated such that it will determine if the y lies above the plane or below it, thus determining from which model the output is produced. Figure 1.2 illustrates this. To state it more formally, there exists a function $a(t)$ in L^2 such that

$$\phi(y) = \langle a, y - \overline{y} \rangle = \int_0^\omega a(t)^T [y(t) - \overline{y}(t)] \, dt \tag{1.35}$$

and ϕ is non-negative on one output set and non-positive on the other. The function $a(t)$ is the *normal function* to the separating hyperplane and ϕ is called the *test function.*

As it is more numerically robust to work away from the boundary of the output sets, we improvise by artificially forcing the sets apart. Suppose that the points closest to each other on the sets are $\overline{y}_0$ and $\overline{y}_1$. Then the line segment between them would be defined by $\overline{y}_0 - \overline{y}_1$ and it would be normal to both sets as the points are the closest to each other in the sets. Using this, we can approximate the hyperplane.

We note here that a larger multiple of the minimum energy detection signal is still proper. More specifically, if v is proper, than $(1+c)v$ is also proper [30, 39], where c is some positive constant. Intuitively, if v pushes the output sets apart, then a detection signal larger than v will also push them apart. Any proper signal can be used to find the hyperplane.

We force the sets further apart by shrinking the noise components. This is equivalent to increasing the detection signal. To see this, consider the following system, where $0 \leq \epsilon \leq 1$:

$$\dot{x} = Ax + B\hat{v} + M\epsilon\mu \tag{1.36a}$$

$$y = Cx + N\epsilon\mu. \tag{1.36b}$$

Multiplying this model by $\frac{1}{\epsilon}$ and letting $z = \frac{x}{\epsilon}$, $w = \frac{y}{\epsilon}$, and $\delta = \frac{1}{\epsilon}$, we get

$$\dot{z} = Az + B\delta\hat{v} + M\mu \tag{1.37a}$$

$$w = Cz + N\mu \tag{1.37b}$$

which is the original model with the larger detection signal and no modification of the noise.

Thus, we will use the model:

$$\dot{x}_i = A_i x_i + B_i\hat{v} + M_i\epsilon\mu_i \tag{1.38a}$$

$$y_i = C_i x_i + N_i\epsilon\mu_i. \tag{1.38b}$$

As we want the points on the boundaries of the closures of the sets which are closest to each other, we change the noise measure to

$$\| \mu_i \|^2 \leq 1, \; i = 0, 1. \tag{1.39}$$

To compute the normal to the separating plane, we minimize $\| y_0 - y_1 \|^2$ subject to (1.38) and (1.39). Then the difference of the solutions can be normalized and is normal to the hyperplane. We use the midpoint of the line segment between $\overline{y}_0$ and $\overline{y}_1$ as the point needed to define the hyperplane.

Model Identification Algorithm

1. Let $\hat{v}$ be the minimum energy proper detection signal from the MEDS algorithm. (SOCS)

2. Choose a value for $\epsilon < 1$.

3. Perform constrained optimization (SOCS)

$$\min \parallel y_0 - y_1 \parallel^2 \tag{1.40}$$

   subject to the constraints

$$\begin{aligned} \dot{x}_i &= A_i x_i + B_i \hat{v} + M_i \epsilon \mu_i && (1.41a) \\ y_i &= C_i x_i + N_i \epsilon \mu_i && (1.41b) \\ \dot{q}_i &= \mu_i^T \mu_i, \ q_i(0) = 0, \ q_i(\omega) \leq 1 && (1.41c) \end{aligned}$$

   for $i = 0, 1$.

4. Let $y_{0,\epsilon}$ and $y_{1,\epsilon}$ be the closest points computed by the optimization.

5. Compute $a_\epsilon(t)$ be the separating hyperplane (MATLAB)

$$a_\epsilon = \frac{\overline{y}_{0,\epsilon} - \overline{y}_{1,\epsilon}}{\parallel \overline{y}_{0,\epsilon} - \overline{y}_{1,\epsilon} \parallel}.$$

6. Compute $\overline{y}_\epsilon(t)$, the point on the separating hyperplane, as the midpoint, of the line segment connecting $\overline{y}_{0,\epsilon}$ and $\overline{y}_{1,\epsilon}$ (MATLAB)

$$\overline{y}_\epsilon = \frac{\overline{y}_{0,\epsilon} + \overline{y}_{1,\epsilon}}{2}.$$

7. Let $\phi_\epsilon(z) = \langle a_\epsilon, z - \overline{y}_\epsilon \rangle$ be the test function. Then

$$\begin{aligned} \phi_\epsilon(y_0) &= \langle a_\epsilon, y_0 - \overline{y}_\epsilon \rangle > \underline{\epsilon} \\ \phi_\epsilon(y_1) &= \langle a_\epsilon, y_1 - \overline{y}_\epsilon \rangle < \overline{\epsilon} \end{aligned}$$

   where y_i is an unknown output from model i, $i = 0, 1$.

8. Suppose a known output from model 0 produces a positive test function value. Then if an unknown output produces a positive test function value, the unknown output derives from model 0. If the unknown output produces a negative test function value, it derives from model 1. If a zero test function value is produced, an error has occurred and a smaller value of ϵ should be selected.

Numerical error increases due to division by small numbers, potentially effecting the results of the hyperplane test. This problem can be avoided the calculations are done to a sufficient level of accuracy. However, a suitable value of ϵ will also do. For a more detailed discussion of the choice of ϵ, see Chapter 3 in [30].

Multiple models

It is possible to find a detection signal with the MEDS algorithm which works for multiple models. Moreover, the model identification algorithm is applicable to multiple models, as well. When there are more than two models the output, y, may not come from either model being compared. Then the test becomes

$$\text{if } y \in \bigcup_{p=1}^{m} \mathcal{A}_p(v), \text{ then } \begin{cases} \phi_{i,j} & < \quad \overline{\epsilon}_{i,j} \Rightarrow y \notin \mathcal{A}_i(v) \\ \phi_{i,j} & > \quad \underline{\epsilon}_{i,j} \Rightarrow y \notin \mathcal{A}_j(v) \end{cases}.$$

1.4 Outline of Thesis

In the chapter that follows, we explore the possibility of making the detection decision early. Then in Chapter 3, we introduce the delayed detection problem and look at solutions to it using the Method of Steps. With this background in the delayed detection problem, in Chapter 4, we look at a solution to the delayed detection problem by reformulating the problem into a partial differential equation (PDE) and then approximating the solution to the PDE with central differences. We look at another approximation in Chapter 5. This approximation adapts a spline method found in a paper by Ito and Kappel [34]. Then in Chapter 6, we look at some numerical examples. In the final chapter, we review some potentially interesting extensions to this work and give our conclusions.

1.5 Contributions of Thesis

The work done as part of this thesis appears in the following papers :

- S. Campbell, K. Drake, R. Nikoukhah, and F. Delebecque, *Rapid Multi-Model Identification In Systems With Delays*, 3rd IFAC Workshop on Time Delay Systems (TDS 2001), Dec. 8-10, Sante Fe, New Mexico, 2001, pp. 296-301.

- S. Campbell, K. Drake, R. Nikoukhah, Early Decision Making When Using Proper Auxillary Signals, Proc. IEEE Conference on Decision and Control, Dec.10-13, Las Vegas, Nevada, 2002, pp. 1832-1837.

- S. Campbell, K. Drake, R. Nikoukhah, *Auxillary System Design for Multimodel Identification in Systems with Multiple Delays*, Proc. 10th Mediterranean Conference on Control and Automation, Lisbon, July 7-11, 2002.

- S. Campbell, K. Drake, R. Nikoukhah, *Analysis of Spline Based Auxiliary Signal Design for Failure Detection in Delay Systems*, Paper accepted by IEEE CSMC 2003.

- Kimberly J. Drake, *An Examination of the Potential of Using the World Wide Web to Create User Interfaces* , M&CT-TECH-00-013, December 2000, Mathematics and Computing Technology, A Division of the Boeing Co.

In addition, a number of other computational studies on systems without delays which are not discussed in this thesis are included in Chapter 4 of [12].

Chapter 2

Early Detection

2.1 What is Early Detection and Why Bother?

We have,

$$\dot{x}_i = A_i x_i + B_i \hat{v} + M_i \mu_i \quad (2.1a)$$

$$y_i = C_i x_i + N_i \mu_i \quad (2.1b)$$

for $i = 0 \dots m$, for $t \geq 0$ where x_i, y, v, μ_i are the system states, output, detection signal and noise, respectively. A_i, B_i, C_i, M_i, N_i are matrices of appropriate dimension. v and μ_i are taken to be in $L^2[0, \omega] = L^2$. This implies that x_i and y_i are in L^2, as well. We also assume that v is proper in the sense defined in Section 1.3 and that the output sets of the two models using this signal are distinct. According to the model identification algorithm of Section 1.3.2, we can then construct the hyperplane test function

$$\phi_{i,j}(y) =< y - \bar{y}_{i,j}, a >= \int_0^\omega a_{i,j}(t)^T (y(t) - \bar{y}_{i,j}(t))\ dt$$

where a is a normal to a hyperplane between the output sets, $\bar{y}$ is a point on the hyperplane, and $[0, \omega]$ is the test interval. As a and $\bar{y}$ are computed offline, the hyperplane test is the calculation of a single integral which can be computed as the output, y, becomes available. Thus, the hyperplane test can be carried out in real time.

The question we wish to address in this chapter is whether it is necessary to wait until the end of the test interval to make the model identification decision. By developing a modified hyperplane test, based on the original test, we will show that it is possible to make an early decision and computational tests will show that it is possible to reduce the time for the hyperplane test by about 25% on average and still maintain perfect model identification.

2.2 Analytical Breakdown of Problem

Assume we have the test function as described in Section 1.3.2

$$\phi_{i,j}(y) = < y - \bar{y}_{i,j}, a >= \int_0^{\omega} a_{i,j}(t)^T (y(t) - \bar{y}_{i,j}(t))\ dt$$

where the test interval is $[0, \omega]$, $a_{i,j}(t) \in L^2[0, \omega]$ is the normal to the separating hyperplane and $\bar{y}_{i,j}$ is a point on the plane. We also assume that if $\phi_{i,j}(y) > \underline{\epsilon}_{i,j} > 0$ we know the output did *not* originate from model j. If $\phi_{i,j}(y) < \bar{\epsilon}_{i,j} < 0$, we know the output is *not* from model i.

In the worst case, we may have to wait until the end of the interval in order to conclusively determine from which model the output originates. However, it is possible that we might be able to make this determination sooner. Let $c \in [0, \omega]$. Let $\phi_{i,j}(y) = \psi_{i,j}(y, c) + \Delta_{i,j}(y, c)$ where

$$\psi_{i,j}(y, c) = \int_0^{c} a_{i,j}(t)^T (y(t) - \overline{y}_{i,j}(t))\ dt$$

and

$$\Delta_{i,j}(y, c) = \int_c^{\omega} a_{i,j}(t)^T (y(t) - \overline{y}_{i,j}(t))\ dt.$$

Notice that $\psi_{i,j}$ is the value of the hyperplane test function calculated until time c, as the output y becomes available, and $\Delta_{i,j}$ is that part which is left to be calculated over the remaining test period $[c, \omega]$.

We have assumed that if $\phi_{i,j}(y) < \bar{\epsilon}_{i,j}$, we know the output is *not* from model i. But, if

$$\phi_{i,j}(y) = \psi_{i,j}(y, c) + \Delta_{i,j}(y, c) \quad < \quad \bar{\epsilon}_{i,j} \quad \Rightarrow y \notin \mathcal{A}_i(v),$$

then

$$\psi_{i,j}(y, c) \quad < \quad \bar{\epsilon}_{i,j} - \Delta_{i,j}(y, c) \quad \Rightarrow y \notin \mathcal{A}_i(v).$$

The idea behind our early decision test is to devise an estimate for $\Delta_{i,j}(c)$ which will act as a threshold. Once $\psi_{i,j}$ passes the threshold, the decision making process will be complete.

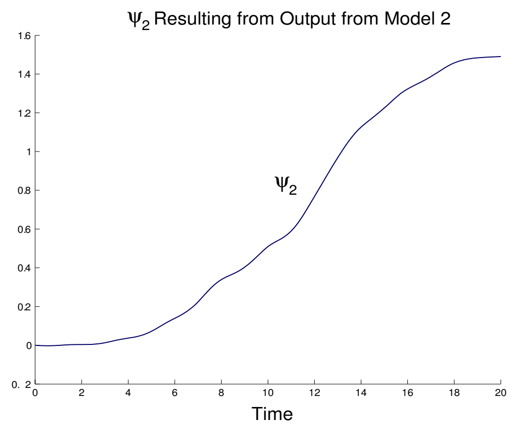

Figure 2.1: An example of $\psi_{i,j}$

In a worst case, we may not be able to make a decision until the end of the test interval. However, it is likely that we will be able to reach a decision earlier. Figure 2.1 shows $\psi_{i,j}$ from an example we will discuss later in this chapter.

Let $\mathcal{L}_p(f) = \int_0^T e^{A_p(t-s)} f(s)ds$ be the solution of $\dot{z} = A_p z + f$ with $z(0) = 0$. Let $x_p(0)$ be the free initial condition for model p. Using this notation, we have

$$y = C_p\mathcal{L}_p(B_p v) + (C_p\mathcal{L}_p M_p + N_p)\mu_p + C_p e^{A_p} x_p(0). \tag{2.4}$$

Then Δ can be rewritten as

$$\Delta_{i,j}(c) = \int_c^\omega a_{i,j}^T (C_p\mathcal{L}_p(B_p v) + C_p e^{A_p s} x_p(0)_p + (C_p\mathcal{L}M + N_p)\mu_p - \bar{y}_{i,j}) \;\; ds. \tag{2.5}$$

Notice that in estimating $\Delta_{i,j}(y, c)$ a smaller estimate is available by grouping as many terms together as possible. That is,

$$\begin{aligned} \Delta_{i,j}(c) \;\; &= \;\; \int_c^\omega a_{i,j}^T (C_p\mathcal{L}_p(B_p v) \;\; ds + \int_c^\omega a_{i,j}^T (C_p e^{A_p s} x_p(0)) \;\; ds \\ &+ \int_c^\omega a_{i,j}^T (C_p\mathcal{L}M + N_p)\mu_p \;\; - \bar{y}_{i,j}) \;\; ds. \end{aligned}$$

Theorem 1. *Suppose that v is a proper m-model auxiliary signal. Suppose that $y \in$*

$\bigcup_{p=1}^{m} \mathcal{A}_p(v)$. *Let*

$$\begin{aligned}
\delta_{i,j}(c) &= \left(\int_c^\omega |a_{i,j}|^2\ ds\right)^{\frac{1}{2}} = \|a_{i,j}\|_{L^2[c,\omega]} && (2.7a)\\
K_i(c) &= \|C_i\mathcal{L}_iM_i + N_i\|_{L^2[c,\omega]} && (2.7b)\\
\gamma_{i,j,k}(c) &= \left|\int_c^\omega a_{i,j}^T(s)\,(C_p\mathcal{L}_pB_pv(s) - \bar{y}_{i,j}(s))\,ds\ \right| && (2.7c)\\
\theta_p(c) &= \left(\int_c^\omega |C_pe^{A_ps}Q^{-\frac{1}{2}}|^2\ ds\right)^{\frac{1}{2}}. && (2.7d)
\end{aligned}$$

Then

1. *If for some* $0 < t < \omega$ *we have that*

$$\psi_{i,j}(y,t) > \underline{\epsilon}_{i,j} + \max_p[\gamma_{i,j,p}(t) + (K_p(t) + \theta_p(t))\delta_{i,j}(t)],\ \textit{then}\ \ y \notin \mathcal{A}_j(v)$$

2. *If for some* $0 < t < \omega$ *we have that*

$$\psi_{i,j}(y,t) < \bar{\epsilon}_{i,j} - \max_p[\gamma_{i,j,p}(t) + (K_p(t) + \theta_p(t))\delta_{i,j}(t)],\ \ \textit{then}\ \ y \notin \mathcal{A}_i(v).$$

Proof. We know that

$$\text{if } y \in \bigcup_{k=1}^{m} \mathcal{A}_p(v),\ \text{then} \begin{cases} \phi_{i,j} & < \quad \bar{\epsilon}_{i,j} \Rightarrow y \notin \mathcal{A}_i(v) \\ \phi_{i,j} & > \quad \underline{\epsilon}_{i,j} \Rightarrow y \notin \mathcal{A}_j(v). \end{cases}$$

Suppose $y \in \mathcal{A}_p(v)$ but we do not know what p is. Then Δ can be rewritten in the form of (2.5). In estimating $\Delta_{i,j}(y,c)$ a smaller estimate is available by grouping as many terms together as possible. That is,

$$\begin{aligned}
\Delta_{i,j}(y,c) &= \int_c^\omega a_{i,j}^T(C_p\mathcal{L}_p(B_pv) - \bar{y}_{i,j})\ \ ds + \int_c^\omega a_{i,j}^T(C_pe^{A_ps}x_p(0)_p)\ \ ds \\
&\quad + \int_c^\omega a_{i,j}^T(C_p\mathcal{L}M + N_p)\mu_p\ \ ds \\
&\leq \|\int_c^\omega a_{i,j}^T(C_p\mathcal{L}_p(B_pv) - \bar{y}_{i,j})\ \ ds\| + \|\int_c^\omega a_{i,j}^T(C_pe^{A_ps}x_p(0)_p)\ \ ds\| \\
&\quad + \|\int_c^\omega a_{i,j}^T(C_p\mathcal{L}_pM + N_p)\mu_p\ ds\| \\
&\leq \gamma_{i,j,p} + \|\int_c^\omega a_{i,j}^T(C_pe^{A_ps}x_p(0)_p)\ \ ds\| + \|\int_c^\omega a_{i,j}^T(C_p\mathcal{L}M + N_p)\mu_p\ \ ds\| \\
&\leq \gamma_{i,j,p} + \left(\int_c^\omega |a_{i,j}^T|^2\ ds\right)^{\frac{1}{2}}\left[\left(\int_c^\omega |(C_pe^{A_ps}x_p(0))|^2\ ds\right)^{\frac{1}{2}}\right. \\
&\quad \left. + \left(\int_c^\omega |(C_p\mathcal{L}M + N_p)\mu_p|^2\ ds\right)^{\frac{1}{2}}\right] \text{ by the Cauchy-Schwartz inequality for } L^2 \\
&\leq \gamma_{i,j,p} + \delta_{i,j}\,(\theta_p + K_p)\ \text{ since } \|\mu\| \leq 1 \text{ and } |Q^{-\frac{1}{2}}x_p(0)| < 1 \text{ by the noise measure} \\
&\leq \max_p[\gamma_{i,j,p} + \delta_{i,j}\,(\theta_p + K_p)] \text{ since we don't know what } p \text{ is}
\end{aligned}$$

Then we have

$$\phi_{i,j}(y) = \psi_{i,j}(y,c) + \Delta_{i,j}(y,c) \quad < \quad \overline{\epsilon}_{i,j} \quad \Rightarrow y \notin \mathcal{A}_i(v)$$

so that

$$\psi_{i,j}(y,c) \quad < \quad \overline{\epsilon}_{i,j} - \Delta_{i,j}(y,c) \quad \Rightarrow y \notin \mathcal{A}_i(v).$$

But, since $\Delta_{i,j}(y,c) < \max_p[\gamma_{i,j,p} + \delta_{i,j}(c)\,(\theta_p(c) + K_p)]$, then

$$\overline{\epsilon}_{i,j} - \max_p[\gamma_{i,j,p} + \delta_{i,j}(c)\,(\theta_p(c) + K_p)] \ < \ \overline{\epsilon}_{i,j} - \Delta_{i,j}(y,c).$$

Thus, if

$$\psi_{i,j}(y,c) < \overline{\epsilon}_{i,j} - \Delta_{i,j}(y,c) \quad \Rightarrow y \notin \mathcal{A}_i(v),$$

then

$$\psi_{i,j}(y,c) \ < \ \overline{\epsilon}_{i,j} - \max_p[\gamma_{i,j,p} + \delta_{i,j}(c)\,(\theta_p(c) + K_p)] \quad \Rightarrow y \notin \mathcal{A}_i(v).$$

Similarly, we have

$$\phi_{i,j}(y) = \psi_{i,j}(y,c) + \Delta_{i,j}(y,c) \quad > \quad \underline{\epsilon}_{i,j} \quad \Rightarrow y \notin \mathcal{A}_j(v)$$

so

$$\psi_{i,j}(y,c) \quad > \quad \underline{\epsilon}_{i,j} - \Delta_{i,j}(y,c) \quad \Rightarrow y \notin \mathcal{A}_j(v).$$

But, since

$$\begin{aligned} \underline{\epsilon}_{i,j} - \Delta_{i,j}(y,c) &\leq \underline{\epsilon}_{i,j} + |\Delta_{i,j}(y,c)| \\ &\leq \underline{\epsilon}_{i,j} + \max_p[\gamma_{i,j,p} + \delta_{i,j}(c)\,(\theta_p(c) + K_p)] \end{aligned}$$

then

$$\psi_{i,j}(y,c) > \underline{\epsilon}_{i,j} + \max_p[\gamma_{i,j,p} + \delta_{i,j}(c)\,(\theta_p(c) + K_p)] \quad \Rightarrow y \notin \mathcal{A}_j(v).$$

$\diamondsuit$

2.3 Algorithm

The theorem of the previous section shows that early decision making is possible. In this section, we provide the algorithm for the early decision test. We use standard numerical packages in the calculations. As the quantities involve functions, we use differential equations in the calculations.

Calculating $\delta_{i,j}(c) = \|a_{i,j}\|_{L^2[c,\omega]}$

Let's consider the function $g(c) = \|f(t)\|^2_{L^2[c,\omega]} = \int_c^\omega |f(t)|_2^2\, dt$ for $c \in [0,\omega]$. First, we note that g is monotonically decreasing on the interval $[0,\omega]$ and we note that $g(0) = \|f\|^2_{L^2[c,\omega]}$ while $g(\omega) = 0$. Then given values of the function f, it is possible to compute $g(c)$ by solving

$$\dot{g} = -|f(c)|_2^2, \quad g(0) = \|f\|_2^2, \quad 0 \leq t \leq \omega \tag{2.14}$$

or

$$\dot{g} = -|f(c)|_2^2, \quad g(\omega) = 0, \quad 0 \leq t \leq \omega. \tag{2.15}$$

As we have $a_{i,j}$ precalculated from the model identification algorithm, we can use (2.15) to calculate the value $\delta_{i,j}(c) = \|a_{i,j}\|_{L^2[c,\omega]}$. That is, we solve

$$\begin{aligned} \dot{g_0} &= -a_{i,j}^T a_{i,j} \quad g(\omega) = 0 \\ \delta_{i,j} &= g_0^{\frac{1}{2}}. \end{aligned}$$

Calculating $\gamma_{i,j,k}(c) = \left| \int_c^\omega a_{i,j}^T(s)\left(C_p\mathcal{L}_pB_pv(s) - \bar{y}_{i,j}(s)\right) ds \right|$

Using a similar technique as above, we can solve the following boundary value problem to find $\gamma_{i,j,k}(c)$. As

$$\gamma_{i,j,k}(c) = \left| \int_c^\omega a_{i,j}^T(s)\left(C_p\mathcal{L}_pB_pv(s) - \bar{y}_{i,j}(s)\right) ds \right|,$$

we solve

$$\begin{aligned} \dot{g}_{1p}(t) &= A_pg_{1p}(t) + B_pv(t), \quad g_{1p}(0) = 0 \\ \dot{g}_{2p}(t) &= a_{i,j}^T(s)\left(C_pg_1(t) - \bar{y}_{i,j}(s)\right), \quad g_{2p}(\omega) = 0 \\ \gamma_{i,j,i}(t) &= |g_{2p}|. \end{aligned}$$

Calculating $\theta_p(c) = \left(\int_c^\omega |C_pe^{A_ps}P_p^{-\frac{1}{2}}|^2 \ ds \right)^{\frac{1}{2}}$

The following initial value problem determines $\theta_p(c)$:

$$\begin{aligned} \dot{g}_{3p}(t) &= |C_pe^{A_pt}P_p^{-\frac{1}{2}}|^2, \quad g_{3p}(\omega) = 0 \\ \theta(t) &= g_{3p}^{\frac{1}{2}}(t). \end{aligned}$$

Calculating $K_p(c) = \|C_p\mathcal{L}_pM_p + N_p\|_{L^2[c,\omega]}$

The K_p term is a bit more complicated. First, we note that separating the M_p and N_p terms will make little difference computationally. Thus, we estimate $\|C_p\mathcal{L}_pM_p\|_{L^2[c,\omega]}$ and $\|N_p\|_{L^2[c,\omega]}$ separately. First,

$$\begin{aligned} \|N_p\mu\|_{L^2[c,\omega]} &= \left(\int_c^\omega |N_p|_2^2|\mu|_2^2 \ dt \right)^{\frac{1}{2}} \\ &\le |N_p|_2 \left(\int_c^\omega |\mu|_2^2 \ dt \right)^{\frac{1}{2}} \\ &\le |N_p|_2\|\mu\|_{L^2[0,\omega]} \\ &\le |N_p|_2 \ \text{ since } \ \|\mu\|_{L^2[0,\omega]} < 1. \end{aligned}$$

As N_p is constant, $|N_p|_2$ is very easy to calculate. We can estimate $\|N_p\mu\|_{L^2[c,\omega]}$ by finding $|N_p|_2$.

We also need to calculate estimates for $\|C_p\mathcal{L}_p(M_p\mu)\|_{L^2[c,\omega]}$. First we show that there exists $\widetilde{M}_p$ such that,

$$\|C_p\mathcal{L}_p(M_p\mu)\|_{L^2[c,\omega]} \leq \widetilde{M}_p(t)\|\mu\|_{L^2[c,\omega]}.$$

We know that

$$\|\mathcal{L}_p(M_p\mu)(t)\|_{L^2[c,\omega]} = \left(\int_c^\omega |C_p\mathcal{L}_p(M_p\mu)(t)|_2^2\ dt\right)^{\frac{1}{2}}.$$

Lets consider a particular $t \in [0,\omega]$. Now,

$$\begin{aligned}
|C_p\mathcal{L}_p(M_p\mu)(t)|_2 &\leq \left|\int_0^t C_p e^{A_p(t-s)}M_p\mu(s)\ ds\right|_2 \quad \text{for } \ t\in[0,\omega] \\
&\leq \int_0^t |C_p e^{A_p(t-s)}M_p|_2|\mu(s)|_2\ ds \\
\text{by Cauchy-Schwartz Inequality} \quad &\leq \left(\int_0^t |C_p e^{A_p(t-s)}M_p|_2^2\ ds\right)^{\frac{1}{2}}\left(\int_0^t |\mu(s)|_2^2\ ds\right)^{\frac{1}{2}}.
\end{aligned}$$

Now, let us consider $|C_p e^{A_p(t-s)}M_p|_2$. Since $s \leq t$ are both in $[0,\omega]$, we know that $t-s \in [0,\omega]$, so $|C_p e^{A_p(t-s)}M_p|_2 = |C_p e^{A_p t}M_p|_2$. Also, there exists an $\overline{M}_p(t)$ such that

$$|C_p e^{A_p t}M_p|_2 \leq \overline{M}_p(t) \ \text{ for all } \ t \in [0,\omega].$$

Thus,

$$\begin{aligned}
|\mathcal{L}_p(M_p\mu)(t)|_2 &\leq \left(\int_0^t \overline{M}_p(s)^2\ ds\right)^{\frac{1}{2}}\left(\int_0^t |\mu(s)|_2^2\ ds\right)^{\frac{1}{2}} \\
&= \left(\int_0^t \overline{M}_p(s)^2\ ds\right)^{\frac{1}{2}}\|\mu\|_{L^2[0,t]} \ \text{ for } \ t\in[0,\omega].
\end{aligned}$$

Then letting $\left(\widetilde{M}_p(t)\right)^{\frac{1}{2}} = \left(\int_0^t \overline{M}_p(s)^2\ ds\right)^{\frac{1}{2}}$, $\widetilde{M}_p(t)$ will satisfy the differential equation,

$$\begin{aligned}
\dot{\widetilde{M}}_p(t) &= \overline{M}_p^2 \\
\widetilde{M}_p(0) &= 1
\end{aligned}$$

where $\overline{M}_p = |C_p e^{A_p t}M_p|_2$. Thus, we have for a particular t that

$$|C_p\mathcal{L}_p(M_p\mu)(t)|_2 \ \leq \ \left(\widetilde{M}_p(t)\right)^{\frac{1}{2}}\|\mu\|_{L^2[0,\omega]} \ \text{ since } \ t\in[0,\omega].$$

Now, we go back and reconsider $\|C_p\mathcal{L}_p(M_p\mu)(t)\|_{L^2[c,\omega]}$. We have

$$
\begin{aligned}
\|C_p\mathcal{L}_p(M_p\mu)(t)\|_{L^2[c,\omega]} &= \left(\int_c^\omega |C_p\mathcal{L}_p(M_p\mu)(t)|_2^2 \; dt\right)^{\frac{1}{2}} \\
&\leq \left(\int_c^\omega \left(\widetilde{M}_p(t)^{\frac{1}{2}}\|\mu\|_{L^2[0,\omega]}\right)^2 \; dt\right)^{\frac{1}{2}} \\
&= \left(\int_c^\omega \widetilde{M}_p(t)\|\mu\|^2_{L^2[0,\omega]} \; dt\right)^{\frac{1}{2}} \\
&= \|\mu\|_{L^2[0,\omega]}\left(\int_c^\omega \widetilde{M}_p(t) \; dt\right)^{\frac{1}{2}}.
\end{aligned}
$$

Note that to calculate $\left(\int_c^\omega \widetilde{M}_p(t) \; dt\right)^{\frac{1}{2}}$, we can solve

$$
\begin{aligned}
\dot{\widetilde{\widetilde{M}}}_p(t) &= -\widetilde{M}_p \\
\dot{\widetilde{M}}_p(t) &= \overline{M}_p^2 \\
\widetilde{M}_p(0) &= 1 \\
\widetilde{\widetilde{M}}_p(\omega) &= 0
\end{aligned}
$$

where $\overline{M}_p = |C_p e^{A_p t} M_p|_2$. In order to calculate this, we need $\overline{M}_p(t)$. There are several ways one might do this. For moderately sized problems, MATLAB offers several functions useful for calculating this value. The MATLAB function expm.m uses a Padé approximation to calculate $e^{A_p t}$. expm2.m calculates the value of $e^{A_p t}$ using a Taylor series. Also, MATLAB provides expm3.m which calculates $e^{A_p t}$ by first diagonalizing $A_p t$. Once the value of $e^{A_p t}$ has been found, norm.m and normest.m can be used to find the value of $|C_p e^{A_p t} M_p|_2$.

Once the above quantities have been found, we can calculate K_p using

$$
\begin{aligned}
\dot{\widetilde{M}}_p(t) &= \overline{M}_p^2, \;\; \widetilde{M}_p(0) = 1 \\
\dot{\widetilde{\widetilde{M}}}_p(t) &= -\widetilde{M}_p, \;\; \widetilde{\widetilde{M}}_p(\omega) = 0 \\
K_p(t) &= |N_p| + \widetilde{\widetilde{M}}_p^{\frac{1}{2}}.
\end{aligned}
$$

To summarize and state the above in a concise algorithm, we have the following:

1. **Estimate $\mathbf{\overline{M}_p}$ for $\mathbf{p = 0...m}$.**

   Calculate $\overline{M}_p = |e^{A_p(t)}|^2$.

2. **Estimate $\mathbf{K_p}$ for $\mathbf{p = 0...m}$.**

   Calculate $K_p = \|C_p\mathcal{L}_pM_p + N_p\|_{L^2[c,\omega]}$ by calculating $\widetilde{M}_p(t) = \int_0^t \overline{M}_p(s)^2\ ds\ \widetilde{\widetilde{M}}_p = \int_c^\omega \widetilde{M}_p(t)\ dt$.

$$\begin{aligned}\dot{\widetilde{M}}_p &= \overline{M}_p^2, \quad \widetilde{M}_p(0) = 0\\ \dot{\widetilde{\widetilde{M}}}_p &= -\widetilde{M}_p, \quad \widetilde{\widetilde{M}}_p(\omega) = 0\\ K_p(t) &= |N_p| + \widetilde{\widetilde{M}}_p^{\frac{1}{2}}.\end{aligned}$$

3. **Estimate $\gamma_{\mathbf{i,j,p}}$ for $\mathbf{p = 0...m}$.**

   Calculate $\gamma_{i,j,k}(c) = \left|\int_c^\omega a_{i,j}^T(s)\left(C_p\mathcal{L}_pB_pv(s) - \bar{y}_{i,j}(s)\right) ds\right|$ using

$$\begin{aligned}\dot{g}_{1p}(t) &= A_pg_{1p}(t) + B_pv(t), \quad g_{1p}(0) = 0\\ \dot{g}_{2p}(t) &= a_{i,j}^T(s)\left(C_pg_{1p}(t) - \bar{y}_{i,j}(s)\right), \quad g_{2p}(\omega) = 0\\ \gamma_{i,j,p}(t) &= |g_{2p}|.\end{aligned}$$

4. **Estimate $\theta_{\mathbf{p}}(\mathbf{c})$ for $\mathbf{p = 0...m}$.**

   Calculate $\theta_p(c) = \left(\int_c^\omega |C_pe^{A_ps}P_p^{-\frac{1}{2}}|^2\ ds\right)^{\frac{1}{2}}$ with

$$\begin{aligned}\dot{g}_{3p}(t) &= |C_pe^{A_pt}P_p^{-\frac{1}{2}}|^2, \quad g_{3p}(\omega) = 0\\ \theta(t) &= g_{3p}^{\frac{1}{2}}(t).\end{aligned}$$

5. **Estimate $\delta_{\mathbf{i,j}}(\mathbf{c})$.**

   Use $a_{i,j}$ to calculate $\delta_{i,j}(c) = \|a_{i,j}\|_{L^2[c,\omega]}$

$$\begin{aligned}\dot{g}_0 &= -a_{i,j}^Ta_{i,j} \quad g(\omega) = 0\\ \delta_{i,j} &= g_0^{\frac{1}{2}}.\end{aligned}$$

6. **Estimate $\widehat{\delta}_{\mathbf{i,j}}(\mathbf{y},\mathbf{t})$.**

   Calculate

$$\widehat{\delta}_{i,j} = \max_p[\gamma_{i,j,p}(t) + (K_p(t) + \theta_p(t))\delta_{i,j}(t)].$$

   Then the early decision test is

(a) If for some $0 < t < \omega$, we have that

$$\psi_{i,j} < \underline{\epsilon}_{i,j} + \widehat{\delta}_{i,j}, \text{ then } \; y \notin \mathcal{A}_j(v).$$

(b) If for some $0 < t < \omega$, we have that

$$\psi_{i,j}(y,t) > \overline{\epsilon}_{i,j} - \widehat{\delta}_{i,j}, \quad \text{then } \; y \notin \mathcal{A}_i(v).$$

Note that if there are only two models, then the test becomes

1. If for some $0 < t < \omega$, we have that

$$\psi_{i,j} < \underline{\epsilon}_{i,j} + \widehat{\delta}_{i,j}, \text{ then } \; y \in \mathcal{A}_i(v).$$

2. If for some $0 < t < \omega$, we have that

$$\psi_{i,j}(y,t) > \overline{\epsilon}_{i,j} - \widehat{\delta}_{i,j}, \quad \text{then } \; y \in \mathcal{A}_j(v).$$

2.4 Computational Tests

In order to evaluate the early decision test, we ran 200 simulations using random noises and initial conditions. We considered the following 2-model example.

$$\begin{aligned}
\dot{x}_0 &= -2x_0 + v + (1 \;\; 0)\mu_0 \\
y &= x_0 + (0 \;\; 1)\mu_0 \\
\dot{x}_1 &= -x_1 + v + (1 \;\; 0)\mu_1 \\
y &= x_1 + (0 \;\; 1)\mu_1 \\
& \quad 0 \leq t \leq 20.
\end{aligned}$$

We use $Q_0 = Q_1 = I$ as the weight on the initial condition. In order to find random initial conditions and noises, we first generated on $[-1,1]$ a set of random coefficients for the first 20 Fourier Sine functions and a random initial condition. We then normalized them on L^2 and R^N, respectively. Next, as we needed initial conditions and 2 noises for each model which would satisfy the noise bound $\mathcal{S}_i(x_i(0),\mu_i) = x_i(0)^T Q_i x_i(0) + \int_0^\omega |\mu_i(t)|^2 \;\; dt < 1$, we generated 3 random coefficients on $[0,1]$ which scale the noises and initial conditions to

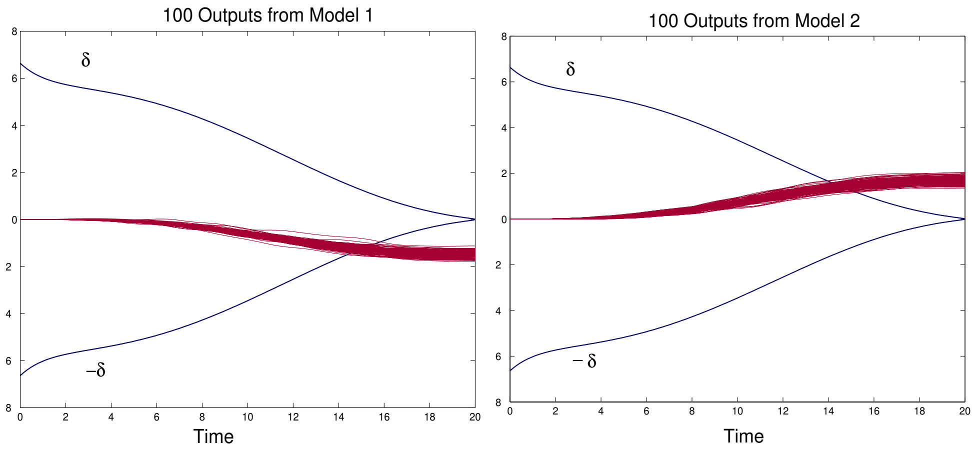

Figure 2.2: When $\psi_{i,j}$, the value of the hyperplane test which is calculated as the output y becomes available, crosses $\widehat{\delta}_{i,j}$ or $-\widehat{\delta}_{i,j}$, the test is complete. In 100 trials on each model, detection takes place around $.75\omega$.

satisfy the noise measure. Figure 2.2 shows the results of the 200 tests where the outputs came from Models 0 and 1 respectively. The smooth curves are $\widehat{\delta}_{i,j}$ and $-\widehat{\delta}_{i,j}$ and the other curves are the 100 graphs of $\psi_{i,j}$. For Model 0, detection time ranged between 14.444 to 15.9722 with an average time of 15.028. Results on Model 1 were similar, with the range between 14.0278 and 15.278 and an average of 14.651. Figure 2.3 gives the actual time distributions for detection. Figure 2.3 shows that over 60% of the trials had resulted in detection by $t = 15$, which is a reduction of 25%. It is worth noting that the jumps in Figure 2.3 are due to the step size in the numerical integrator used in the simulation. It is also worth noting that there is a clustering of detection around $.75\omega$. Detection is unlikely in the beginning of the interval. The end of the interval represents the worst case, which is also unlikely to occur in a random generation of noises.

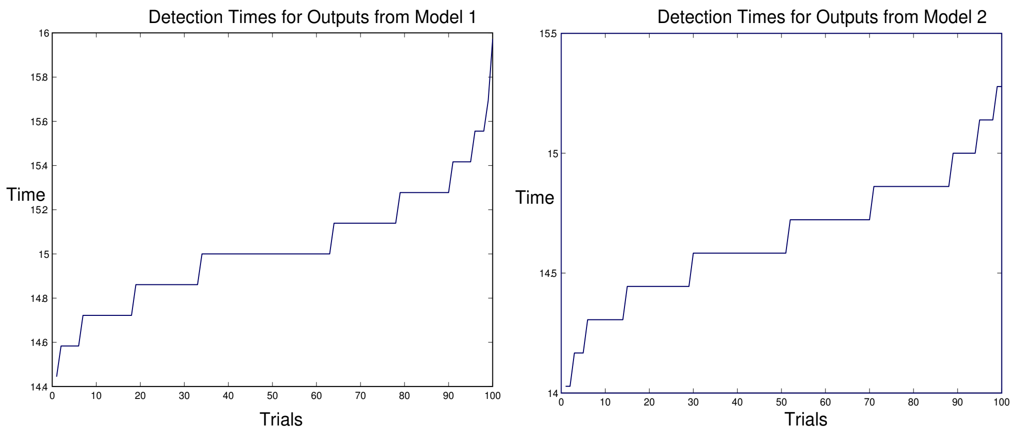

Figure 2.3: Detection time distribution for 100 trials of each model. The figures show that over 60% of the trials had resulted in detection by $t = 15$, which is a reduction of 25%

Chapter 3

True Solutions: The Method of Steps

3.1 Introduction to the Problem

Frequently, we assume that the future of a system is independent of past states and relies only on the present. This is not always a realistic representation of the systems under study. Models which are dependent on the past history of the state are common in many applications. For example, a model of measles spread in a city is given by

$$\dot{S} = -\beta(t)S[2\gamma + s(t-14) - S(t-12)] + \gamma \tag{3.1}$$

with $S(t)$ representing the susceptibles in the population at time t days, γ representing the rate at which individuals enter the population, and $\beta(t)$ representing a function characteristic of the population. An individual exposed at time t is infectious during the period $[t-14, t-12]$ [27].

The circumnutation[1] of sunflowers has been modelled by [27]

$$\dot{\alpha} = -k \int_1^\infty f(\theta)\sin(\alpha(t-\theta-t_0))\ d\theta \tag{3.2}$$

where α represents the angle the top of the plant makes with the vertical. The equation

$$\dot{x} = -\alpha x(t-1)[1 + x(t)]$$

has been used to model the distribution of primes [27]. Later in this chapter, we will present an example of a model used in simulation of Mach number in a wind tunnel [48]. In each of

[1]circumnutation: The successive bowing or bending in different directions of the growing tip of the stems of many plants, especially seen in climbing plants.

these cases, the past history of the system plays a vital role in its future. This past history enters the models in the form of delays, sometimes simply as a single, constant delay, as in the measles model (3.1), or sometimes in a more complex way, as in the model of the sunflower (3.2).

In this chapter, we consider detection on models with delays. The simplest of these models have the form:

$$\begin{aligned} \dot{x} &= Ax(t) + Gx(t-h) + Bv(t) + M\mu(t), \quad t \geq 0 && \text{(3.3a)} \\ x(0) &= \eta, && \text{(3.3b)} \\ x(t) &= \phi(t) \ \text{ a.e. on } \ [-h, 0) && \text{(3.3c)} \\ y(t) &= Cx(t) + N\mu(t), \ t \geq 0. && \text{(3.3d)} \end{aligned}$$

This model illustrates some key similarities and differences between delayed models and models without delays. As usual, we take A, B, M, N, and C to be constant matrices to simplify discussion. G, the coefficient matrix of the delayed term, is also taken to be a constant matrix. Again, μ is the noise and v is the detection signal. They are assumed to be functions in $L^2[0, \omega]$, where ω is the detection horizon. Notice that rather than having just an initial condition, we have the addition of an initial function, $\phi \in L^2[-h, 0)$, describing the past history of the state. In a delayed system, state space is infinite dimensional. This constitutes a significant difference from previously studied models and will require special handling in signal design.

Also, notice that we have no assumption of continuity between the initial function and the initial condition. That is, we do not assume that $\phi(0) = x(0)$. This will have ramifications when we consider the necessary conditions which we use as constraints in finding the detection signal.

There are two ways we can approach the fault detection question on such a model. One way would be to return to first principles, developing a new theory and a new signal design for this class of problems. While this is certainly possible, it is difficult in that we would need to consider optimal control on delayed systems. We need to determine the necessary conditions for (1.18) analytically in order to solve for the detection signal. This is a non-trivial task with a delayed system.

The second way is the time honored mathematical tradition of reducing the problem to one we have done before. By approximating these infinite dimensional systems by finite dimensional ones, we are able to extend previously developed theory to this new class of problems. There are technical and theoretical issues which must be addressed. However, the transition is smooth.

Among the technical and theoretical issues associated with this new class of problem is how the addition of this initial function, ϕ, is handled. Recall the noise measure in Section 1.3

$$\mathcal{S}_i(x_i(0), \mu_i) = x_i(0)^T Q_i x_i(0) + \int_0^{\omega} |\mu_i(t)|^2 \; dt.$$

The initial condition, $x(0)$, is included in the noise measure and treated as a disturbance of the system. Perfect detection is possible only because we bound the noise. Like the initial condition, the initial function will be treated as a disturbance and will contribute to the bounded noise. The noise measure becomes

$$\mathcal{S}_i(x_i(0), \phi_i, \mu_i) = x_i(0)^T Q_i x_i(0) + \int_{-h}^{0} |\phi_i(t)|^2 \; dt + \int_0^{\omega} |\mu_i(t)|^2 \; dt. \tag{3.4}$$

Note that if $\omega < h$ then (3.4) is actually

$$\mathcal{S}_i(x_i(0), \phi_i, \mu_i) = x_i(0)^T Q_i x_i(0) + \int_{-h}^{\omega-h} |\phi_i(t)|^2 \; dt + \int_0^{\omega} |\mu_i(t)|^2 \; dt. \tag{3.5}$$

Also, if $\omega < h$, then

$$\begin{aligned} \dot{x} &= Ax(t) + Gx(t-h) + Bv(t) + M\mu(t), \quad t \geq 0 && \text{(3.6a)} \\ &= Ax(t) + G\phi(t-h) + Bv(t) + M\mu(t), \quad \text{for all } t \in [0, \omega]. && \text{(3.6b)} \end{aligned}$$

Since ϕ is considered a disturbance, then (3.6b) can be rewritten as

$$\dot{x} = Ax(t) + Bv(t) + \tilde{M}\tilde{\mu}(t), \quad t \geq 0$$

where $\tilde{M} = [G \; M]$ and $\tilde{\mu} = \begin{bmatrix} \phi \\ \mu \end{bmatrix}$. The problem has been rewritten without delays and requires no further special care. Thus, we will only consider the case when $\omega > h$.

We approach the problem of reformulating the delay system in three ways. The first is through the Method of Steps, as discussed in Chapter 3. Then we look at reformulating the problem into a PDE and using a central difference approximation, as discussed in Chapter

4. Finally, we look at a direct approximation using splines, as is discussed in Chapter 5. Approximations allow for more complicated models than the Method of Steps. However, the Method of Steps provides a true solution, rather than an approximate one. Thus, solutions using the Method of Steps serve as a basis of comparison and verification of the other two methods.

3.2 True Solutions with the Method of Steps

In this section, we will introduce the Method of Steps. By breaking up the detection horizon into pieces which are the length of the delay and assigning to those pieces a new variable, the delay term can be eliminated from the system. With a few modifications, the theory of previous chapters will apply. Unfortunately, while the Method of Steps provides a true solution, it only works on a small class of problems.

3.2.1 The Basic Problem

A simple illustrative example

In order to motivate the ideas behind the Method of Steps, we first consider a simple example. Suppose we wish to find a continuous solution to the differential delay equation (DDE) given by

$$\dot{x}(t) = x(t-1) \tag{3.8a}$$

$$x(t) = \phi = 1 \text{ for } -1 < t \le 0. \tag{3.8b}$$

Then if (3.8) is to hold for $0 < t \le 1$, we need

$$\begin{aligned} \dot{x}(t) &= x(t-1) \text{ for } 0 < t \le 1 \\ &= x(s) \text{ for } -1 < s \le 0. \end{aligned}$$

Finding x on $0 < t \le 1$ is no more difficult than solving a simple differential equation. That is,

$$\begin{aligned} \dot{x}(t) &= x(t-1) \\ &= 1. \end{aligned}$$

Thus,

$$x(t) = t + c$$

for c a constant. As we require continuity in x and $x(t)$ must satisfy (3.8b), we have

$$\begin{aligned} x(0) &= 1 \\ 0 + c &= 1 \\ c &= 1. \end{aligned}$$

Thus, we have,

$$x(t) = \begin{cases} 1 & -1 < t \leq 0 \\ t+1 & 0 < t \leq 1. \end{cases}$$

This process is repeated, solving for x on $1 < t \leq 2$ with $x(1) = 2$, and so on. In this way, x can be determined for all time. This example illustrates the *Method of Steps*, sometimes called the *Method of Continuation*, by which the solution is extended forward from interval to interval.

Clearly, solving the same differential equations repeatedly over short time intervals may not be desirable computationally. In order to avoid this, we reformulate the problem into a system of higher dimension in which the delayed term has been eliminated. To accomplish this, a new variable is assigned to each interval of length h of the detection horizon, where h is the length of the delay. Then variable z_{i-1} is the past history of variable z_i and the system can be reformulated without the delay term. Using these new variables, a system of equations is set up, allowing the solution for the entire detection horizon to be found simultaneously. A boundary condition $z_{i-1}(h) = z_i(0)$ is included to ensure continuity between these new variables. Figure 3.1 illustrates this.

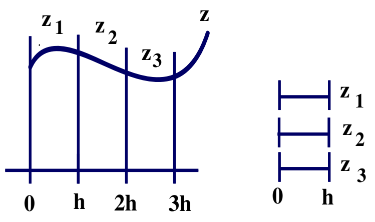

Figure 3.1: Diagram illustrating the Method of Steps: A new variable is assigned to each interval of length h of the detection horizon, where h is the length of the delay. Then variable z_{i-1} is the past history of variable z_i and the system can be reformulated without the delay term. Using these new variables, a system of equations is set up, allowing the solution for the entire detection horizon to be found simultaneously. A boundary condition $z_{i-1}(h) = z_i(0)$ is included to ensure continuity between variables.

Method of steps formalism

Having considered a simple, heuristic example, we present the Method of Steps formalism. Suppose we have two models

$$\begin{aligned} \dot{x}_0 &= A_0x_0(t) + G_0x_0(t-h) + B_0v(t) + M_0\mu_0(t), \quad t \geq 0 && \text{(3.13a)} \\ y(t) &= C_0x_0(t) + N_0\mu_0(t), \;\; t \geq 0. && \text{(3.13b)} \\ x_0(0) &= \eta_0, && \text{(3.13c)} \\ x_0(t) &= \phi_0(t) \;\; \text{a.e. on} \;\; [-h, 0) && \text{(3.13d)} \end{aligned}$$

and

$$\begin{aligned} \dot{x}_1 &= A_1x_1(t) + G_1x_1(t-h) + B_1v(t) + M_1\mu_1(t), \quad t \geq 0 && \text{(3.14a)} \\ y(t) &= C_1x_1(t) + N_1\mu_1(t), \;\; t \geq 0. && \text{(3.14b)} \\ x_1(0) &= \eta_1, && \text{(3.14c)} \\ x_1(t) &= \phi_1(t) \;\; \text{a.e. on} \;\; [-h, 0) && \text{(3.14d)} \end{aligned}$$

where A_i, B_i, M_i, N_i, G_i and C_i are constant matrices and v and μ are in $L^2[0, \omega]$. Notice that both models have the same delay.

Also, we assume that $\omega = \overline{\kappa} h$ with an integer $\overline{\kappa} \geq 1$.

Eliminating y, as discussed in Section 1.3, reduces the systems (3.13) and (3.14) to

$$\begin{aligned}
\dot{x}_0 &= A_0 x_0(t) + G_0 x_0(t-h) + B_0 v(t) + M_0 \mu_0(t), \quad t \geq 0 && \text{(3.15a)}\\
\dot{x}_1 &= A_1 x_1(t) + G_1 x_1(t-h) + B_1 v(t) + M_1 \mu_1(t), \quad t \geq 0 && \text{(3.15b)}\\
0 &= C_0 x_0(t) - C_1 x_1(t) + N_0 \mu_0(t) - N_1 \mu_1(t), \ t \geq 0. && \text{(3.15c)}\\
x_0(t) &= \phi_0(t), \ \ x_1(t) = \phi_1(t) \text{ a.e. on } [-h, 0) && \text{(3.15d)}\\
x_0(0) &= \eta_0, \qquad x_1(0) = \eta_1. && \text{(3.15e)}
\end{aligned}$$

The reformulation of the problem using Method of Steps is given by the following:

Let $\kappa = \overline{\kappa} - 1$. Let $z_{i,j} = x_i(t + jh)$, $\mu_{i,j} = \mu_i(t + jh)$, and $v_i = v(t + jh)$ for t on $[0, h]$ and $j = 1 \ldots \kappa$. For $0 \leq t \leq h$, let $\gamma_i(t) = \phi_i(t - h)$. Then the system is

$$\begin{aligned}
\dot{z}_{0,0} &= A_0 z_{0,0} + G_0 \gamma_0 + B_0 v_0 + M_0 \mu_{0,0}. && \text{(3.16a)}\\
&\vdots && \text{(3.16b)}\\
\dot{z}_{0,\kappa} &= A_0 z_{0,\kappa} + G_0 z_{0,\kappa-1} + B_0 v_\kappa + M_0 \mu_{0,\kappa} && \text{(3.16c)}\\
\hline
\dot{z}_{1,0} &= A_1 z_{1,0} + G_1 \gamma_1 + B_1 v_0 + M_1 \mu_{1,0} && \text{(3.16d)}\\
&\vdots && \text{(3.16e)}\\
\dot{z}_{1,\kappa} &= A_1 z_{1,\kappa} + G_1 z_{1,\kappa-1} + B_1 v_\kappa + M_1 \mu_{1,\kappa} && \text{(3.16f)}\\
\hline
0 &= C_0 z_{0,0} - C_1 z_{1,0} + N_0 \mu_{0,0} - N_1 \mu_{1,0}, \ t \geq 0 && \text{(3.16g)}\\
&\vdots && \text{(3.16h)}\\
0 &= C_0 z_{0,\kappa} - C_1 z_{1,\kappa} + N_0 \mu_{0,\kappa} - N_1 \mu_{1,\kappa}, \ t \geq 0 && \text{(3.16i)}\\
z_{i,0}(0) &= x_i(0) \ \ i = 0, 1 && \text{(3.16j)}\\
z_{i,j}(0) &= z_{i,j-1}(h) \ \ i = 0, 1 \ \ j = 1, \ldots, \kappa. && \text{(3.16k)}
\end{aligned}$$

It should be clear from the above that the size of the problem increases significantly as κ increases which occurs when the delay gets small relative to the detection horizon. In fact, in the next section, we shall see that this increase is even more dramatic, as there will also be an increase in the number of multipliers introduced in finding the necessary conditions to the inner minimum of the MEDS algorithm. Depending on the choice of optimization software, this can significantly increase computational effort. A sparse, direct transcription code like SOCS finds a mesh on the interval $[0,\ h]$ which corresponds to the mesh on $[0,\ \omega]$.

A significant computational difference is only seen if the mesh on $[0,\ h]$ is much finer than would be necessary to resolve the dynamics on $[0,\ \omega]$.

It should be noted that, technically, it is possible to have more than one delay or different delays and use the Method of Steps. Suppose we have delays r_i for $i = 1 \ldots n$ and our detection test is on the interval from $[0, \omega]$. Then by choosing h such that h divides r_i for all i and h divides ω, we can set up a system similar to (3.16) on $[0, h]$. However, solving the problem in this way will *dramatically* increase the dimensions of the computational problem. For example, later in Chapter 6, we will look at an example with delays of -1 and $-.4$ where the detection horizon is $[0, 2]$. In order to solve this problem with the Method of Steps, we would need to choose $h = .1$. This means for *each* model, we would be increasing the number of states from 1 to 20. Thus, on even a simple example with mixed delays, this is a non-trivial restriction and we will not even consider it as a genuine option.

For ease in discussion, we will write the system (3.16) on $[0, h]$ as

$$\begin{aligned}
\dot{\overline{z}} &= \overline{A}\overline{z} + \overline{B}\overline{v} + \overline{M}\overline{\mu} \\
0 &= \overline{C}\overline{z} + \overline{N}\overline{\mu} \\
\overline{z}_i(0) &= \overline{z}_{i-1}(h) \text{ for } \ i = 1 \ldots \kappa, \kappa + 2 \ldots 2\kappa + 2
\end{aligned}$$

where $\overline{r} = \begin{bmatrix} \overline{r}_0 \\ \overline{r}_1 \end{bmatrix}$ for $\overline{r} = \overline{z}, \overline{\mu}$ and $\overline{R} = \text{diag}\{\overline{R}_0, \overline{R}_1\}$ for $\overline{A}, \overline{B}, \overline{M}, \overline{N}$, and $\overline{C}$ with $\overline{B}_i =$ $\text{diag}\{B_i, \ldots, B_i\}$, $\overline{C}_i = \text{diag}\{C_i, \ldots, C_i\}$, $\overline{A}_i = \begin{bmatrix} A_i & 0 & 0 & 0 \\ G_i & A_i & 0 & \ddots \\ 0 & \ddots & \ddots & \ddots \\ 0 & 0 & G_i & A_i \end{bmatrix}$, $\overline{z}_i = \begin{bmatrix} z_{i,0} \\ \vdots \\ z_{i,\kappa} \end{bmatrix}$, $\overline{v} = \begin{bmatrix} v_0 \\ \vdots \\ v_\kappa \end{bmatrix}$,

$\overline{\mu}_i = \begin{bmatrix} \gamma_i \\ \mu_{i,0} \\ \vdots \\ \mu_{i,\kappa} \end{bmatrix}$, $\overline{N}_i = \begin{bmatrix} 0 & N_i & 0 & \ldots \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & N_i \end{bmatrix}$, and $\overline{M}_i = \begin{bmatrix} G_i & M_i & 0 & \ldots \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & M_i \end{bmatrix}$.

Notice that γ_i, which represents the initial function of our original system, is now incorporated in the noise vector and will be treated as a disturbance as previously discussed.

Thus, the noise measure for the system is

$$\mathcal{S}_i(z,\mu) = \overline{z}(0)^T\overline{Q}_i\overline{z}_{i,0}(0) + \int_0^h |\overline{\mu}_i(t)|^2 \;\; dt \tag{3.18}$$

where $\overline{Q}_i = \text{diag}\{Q_i,\dots,Q_i\}$.

3.2.2 Necessary Conditions of the Optimal Control Problem and Proof of Optimality

Having reformulated the system without a delay, we now consider the signal design problem of Section 1.3. More specifically, we must find

$$\min \|v\|^2$$

such that

$$\max_{0\le\beta\le 1}\min_{\overline{z}_i,\overline{\mu}_i} \beta\mathcal{S}_0(\overline{z}_0(0),\overline{\mu}_0) + (1-\beta)\mathcal{S}_1(\overline{z}_1(0),\overline{\mu}_1) \ge 1 \tag{3.19}$$

subject to

$$\begin{aligned}\dot{\overline{z}} &= \overline{A}\overline{z} + \overline{B}\overline{v} + \overline{M}\overline{\mu}\\ 0 &= \overline{C}\overline{z} + \overline{N}\overline{\mu}\\ \overline{z}_i(0) &= \overline{z}_{i-1}(h) \text{ for } \; i = 1\dots\kappa, \kappa+2\dots 2\kappa+2\end{aligned}$$

for $i = 1, 2$.

To create the optimal control problem which we will solve for the detection signal, we must first derive the necessary conditions for the inner minimum of (3.19). Then we will prove the optimality of the conditions we derive.

Deriving necessary conditions of the inner minimum

We need to find the optimality conditions of the following problem. We have

$$\min_{\overline{z},\overline{\mu}} \overline{z}(0)^T\overline{Q}_\beta\overline{z}(0) + \frac{1}{2}\int_0^h \overline{\mu}^T V_\beta \overline{\mu}\; dt \tag{3.21}$$

where $\frac{1}{2}\overline{\mu}^T V_\beta\overline{\mu} = \beta\|\overline{\mu}_0\|^2 + (1-\beta)\|\overline{\mu}_1\|^2$ and $\overline{Q}_\beta = \text{diag}\{\beta\overline{Q}_0, (1-\beta)\overline{Q}_1\}$ subject to

$$\begin{aligned}\dot{\overline{z}} &= \overline{A}\overline{z} + \overline{B}\overline{v} + \overline{M}\overline{\mu}\\ 0 &= \overline{C}\overline{z} + \overline{N}\overline{\mu}\end{aligned}$$

with the boundary conditions $\overline{z}_i(0) = \overline{z}_{i-1}(h)$ for $i = 1 \dots \kappa,\ \kappa+2 \dots 2\kappa+2$ and $z_0(0)$, $z_{\kappa+1}(0)$, $z_\kappa(h)$, $z_{2\kappa+2}(h)$ free.

First, we define some notation for the derivation of the necessary conditions. Note that we drop the bar notation for simplicity in this section. Let

$$\phi(z(0)) \;=\; z(0)^T Q_\beta z(0) = \beta z_0^T(0) Q_0 z_0(0) + (1-\beta) z_1^T(0) Q_1 z_1(0)$$

and

$$\psi(z(0), z(h)) \;=\; P_1 z(0) + P_2 z(h) = \begin{bmatrix} 0 \\ z_1(0) - z_0(h) \\ \vdots \\ z_\kappa(0) - z_{\kappa-1}(h) \\ 0 \\ z_{\kappa+2}(0) - z_{\kappa+1}(h) \\ \vdots \\ z_{2\kappa+2}(0) - z_{2\kappa+1}(h) \end{bmatrix}.$$

Let

$$\begin{aligned} f &= Az + Bv + M\mu \\ g &= Cz + N\mu \\ L &= \frac{1}{2}\mu^T V_\beta \mu. \end{aligned}$$

We find the necessary conditions by adjoining the constraints to the cost with multipliers. That is, we set up

$$J' = \phi + \nu^T \psi + \int_0^h [L + \lambda^T (f - \dot{z}) + \eta^T g] \;\; dt.$$

Letting $H = L + \lambda^T f + \eta^T g$ be the Hamiltonian, we have

$$J' = \phi + \nu^T \psi + \int_0^h [H - \lambda^T \dot{z}] \;\; dt.$$

Using Leibnitz's rule and incrementing J' as a function of increments in z, λ, ν, η, and μ, gives

$$\begin{aligned} dJ' &= (\phi_z^T + \nu^T \psi_{z(0)})\; dz(0) + \nu^T \psi_{z(h)}\; dz(h) + \psi^T d\nu \\ &+ \int_0^h [H_z^T\; \delta z + H_\mu^T\; \delta\mu - \lambda\; \delta\dot{z} + (H_\lambda - \dot{z})^T\; \delta\lambda + H_\eta\; \delta\eta] \;\; dt. \end{aligned}$$

Eliminating the variation in $\dot{z}$ with an integration by parts and using the fact that $dz(0) = \delta z(0)$ and $dz(h) = \delta z(h)$ since there is no variation in initial or final time, we have

$$\begin{aligned} dJ' &= (\phi_z^T + \nu^T \psi_{z(0)} + \lambda(0))\ dz(0) + (\nu^T \psi_{z(h)} - \lambda(h))\ dz(h) + \psi^T d\nu \\ &+ \int_0^h [(H_z^T + \dot{\lambda}^T)\ \delta z + H_\mu^T\ \delta\mu + (H_\lambda - \dot{z})^T\ \delta\lambda + H_\eta\ \delta\eta]\ \ dt. \end{aligned}$$

Setting coefficients of $\delta z, \delta\eta, \delta\mu, d\nu$, and $\delta\lambda$ to 0, we get the expected conditions

$$\begin{aligned} \dot{\overline{z}} &= \overline{A}\overline{z} + \overline{B}\overline{v} + \overline{M}\overline{\mu} \\ \dot{\overline{\lambda}} &= -\overline{A}^T\overline{\lambda} + \overline{C}^T\eta. \\ 0 &= V_\beta\overline{\mu} - \overline{M}^T\overline{\lambda} + \overline{N}^T\eta \\ 0 &= \overline{C}\overline{z} + \overline{N}\overline{\mu} \\ \overline{z}_j(0) &= \overline{z}_{j-1}(h) \quad \text{for}\ \ j = 1\ldots\kappa, \kappa+2\ldots 2\kappa+2. \end{aligned}$$

Now we consider the transversality conditions in detail. We have

$$\phi_z^T + \nu^T\psi_{z(0)} + \lambda(0) = 0 \quad \text{and} \quad \nu^T\psi_{z(h)} - \lambda(h) = 0.$$

Expanding these vectors, we have,

$$\phi_z^T + \nu^T\psi_{z(0)} + \lambda(0) = \left[\begin{array}{c} 2\beta Q_0 z_0(0) + \lambda_0(0) \\ \nu_1 + \lambda_1(0) \\ \vdots \\ \nu_\kappa + \lambda_\kappa(0) \\ \hline \\ 2(1-\beta)Q_1 z_{\kappa+1}(0) + \lambda_{\kappa+1}(0) \\ \nu_{\kappa+2} + \lambda_{\kappa+2}(0) \\ \vdots \\ \nu_{2\kappa+2} + \lambda_{2\kappa+2}(0) \end{array}\right] = 0$$

and

$$\nu^T \psi_{z(h)} - \lambda(h) = \left[\begin{array}{c} -\nu_1 - \lambda_0(h) \\ \vdots \\ -\nu_\kappa - \lambda_{\kappa-1}(h) \\ -\lambda_\kappa(h) \\ \hline \\ -\nu_{\kappa+2} - \lambda_{\kappa+1}(h) \\ \vdots \\ -\nu_{2\kappa+2} - \lambda_{2\kappa+1}(h) \\ -\lambda_{2\kappa+2}(h) \end{array}\right] = 0.$$

Then using the above $2(2\kappa+2)$ equations, we can eliminate ν entirely and finish with the boundary conditions

$$\begin{aligned} 2\beta Q_0 z_0(0) + \lambda_0(0) &= 0 \\ 2(1-\beta)Q_1 z_{\kappa+1}(0) + \lambda_{\kappa+1}(0) &= 0 \\ \lambda_i(0) - \lambda_{i-1}(h) &= 0 \ \text{ for } i = 1\ldots\kappa, \kappa+2\ldots 2\kappa+2 \\ \lambda_j(h) &= 0 \quad \text{for} \quad j = \kappa, 2\kappa+2. \end{aligned}$$

Finally, we have the following necessary conditions:

$$\begin{aligned} \dot{\overline{z}} &= \overline{A}\overline{z} + \overline{B}\overline{v} + \overline{M}\overline{\mu} && \text{(3.32a)} \\ \dot{\overline{\lambda}} &= -\overline{A}^T\overline{\lambda} + \overline{C}^T\eta. && \text{(3.32b)} \\ 0 &= V_\beta\overline{\mu} - \overline{M}^T\overline{\lambda} + \overline{N}^T\eta && \text{(3.32c)} \\ 0 &= \overline{C}\overline{z} + \overline{N}\overline{\mu} && \text{(3.32d)} \\ \overline{\lambda}_j(0) &= \overline{\lambda}_{j-1}(h) \quad \text{for} \quad j = 1\ldots\kappa, \kappa+2\ldots 2\kappa+2 && \text{(3.32e)} \\ 0 &= 2\beta Q_0 z_0(0) + \lambda_0(0) && \text{(3.32f)} \\ 0 &= 2(1-\beta)Q_1 z_{\kappa+1}(0) + \lambda_{\kappa+1}(0) && \text{(3.32g)} \\ \overline{z}_j(0) &= \overline{z}_{j-1}(h) \quad \text{for} \quad j = 1\ldots\kappa, \kappa+2\ldots 2\kappa+2 && \text{(3.32h)} \\ \lambda_j(h) &= 0 \quad \text{for} \quad j = \kappa, 2\kappa+2. && \text{(3.32i)} \end{aligned}$$

We wish to show that the above necessary conditions yield an optimal solution to the problem defined in (3.21).

Theorem 2. *Suppose that* $\overline{z}, \overline{\mu}, \overline{\lambda}, \eta$ *is a solution to (3.21) using the necessary conditions (3.32). Then* $\overline{\mu}$ *is optimal.*

Proof. (For ease in exposition, we drop the bar from the notation in this proof.) Let $J = J_{v,\beta}(z,\mu) = z^T(0)Q_\beta z(0) + \mu^T V_\beta \mu$. We show that $J(\hat{\mu}) \geq J(\mu)$ for every μ satisfying our conditions by showing that the function $f(s) = J(s(z,\mu) + (1-s)(\hat{z},\hat{\mu}))$ has a minimum at $s = 1$ for all (z,μ).

Let $J_0 = \hat{z}^T(0)Q_\beta z(0) + < \hat{\mu}, V_\beta \mu >, J = z^T(0)Q_\beta z(0) + < \mu, V_\beta \mu >$, and $\hat{J} = \hat{z}^T(0)Q\hat{z}(0) + < \hat{\mu}, V_\beta \hat{\mu} >$. Then f is quadratic in s and has a minimum at

$$s = \frac{\hat{J} - J_0}{J - 2J_0 + \hat{J}}.$$

If $J_0 = J$, then $s = 1$. We have

$$\begin{aligned}
J_0 - J &= \hat{z}^T(0)Q_\beta z(0) - z^T(0)Q_\beta z(0) + < V_\beta \mu, \hat{\mu} > - < V_\beta \mu, \mu > \\
&= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + < V_\beta \mu, \hat{\mu} - \mu > \\
&= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + < M^T \lambda - N^T \eta, \hat{\mu} - \mu > \\
&= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + < \lambda, M(\hat{\mu} - \mu) > - < \eta, N(\hat{\mu} - \mu) > \\
&= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + < \lambda, M(\hat{\mu} - \mu) > + < \eta, C(\hat{z} - z) > \\
&= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + < \lambda, M(\hat{\mu} - \mu) > + < C^T \eta, (\hat{z} - z) > \\
&= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + < \lambda, M(\hat{\mu} - \mu) > + < \dot{\lambda} + A^T \lambda, \hat{z} - z > \\
&= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + < \lambda, M(\hat{\mu} - \mu) > + < \dot{\lambda}, \hat{z} - z > \\
&\quad + < A^T \lambda, \hat{z} - z > \\
&= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + < \lambda, M(\hat{\mu} - \mu) > + < \dot{\lambda}, \hat{z} - z > \\
&\quad + < \lambda, A(\hat{z} - z) > \\
&= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + < \lambda, M(\hat{\mu} - \mu) > + < \dot{\lambda}, \hat{z} - z > \\
&\quad + < \lambda, (\dot{\hat{z}} - \dot{z}) - B(\hat{v} - v) - M(\hat{\mu} - \mu) > .
\end{aligned}$$

But $\hat{v} = v$, so we have,

$$\begin{aligned} J_0 - J &= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + <\lambda, M(\hat{\mu} - \mu)> \\ &\quad + <\dot{\lambda}, \hat{z} - z> + <\lambda, (\dot{\hat{z}} - \dot{z})> - <\lambda, M(\hat{\mu} - \mu)> \\ &= (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + <\dot{\lambda}, \hat{z}> + <\lambda, \dot{\hat{z}}> -(<\dot{\lambda}, z> + <\lambda, \dot{z}>). \end{aligned}$$

We eliminate $<\lambda, \dot{\bar{z}}>$ with an integration by parts. That is, we have

$$\begin{aligned} <\lambda, \dot{z}> &= \int_0^h \lambda^T \dot{z}\, dt \\ &= \lambda^T z|_0^h - \int_0^h \dot{\lambda} z\, dt \\ &= \lambda^T z|_0^h - <\dot{\lambda}, z> . \end{aligned}$$

Then we have

$$J_0 - J = (\hat{z}^T(0) - z^T(0))Q_\beta z(0) + <\dot{\lambda}, \hat{z}> + <\lambda, \dot{\hat{z}}> -(<\dot{\lambda}, z> + \lambda^T z|_0^h - <\dot{\lambda}, z>).$$

Consider the first few terms of $\overline{\lambda}^T z|_0^h$. We have

$$\begin{aligned} \overline{\lambda}^T z|_0^h &= \overline{\lambda}_0(h)z_0(h) - \overline{\lambda}_0(0)z_0(0) + \overline{\lambda}_1(h)z_1(h) - \overline{\lambda}_1(0)z_1(0) \\ &+ \overline{\lambda}_2(h)z_2(h) - \overline{\lambda}_2(0)z_2(0) + \overline{\lambda}_3(h)z_3(h) - \overline{\lambda}_3(0)z_3(0) \\ &\qquad\vdots \\ &+ \overline{\lambda}_{\kappa+1}(h)z_{\kappa+1}(h) - \overline{\lambda}_{\kappa+1}(0)z_{\kappa+1}(0) + \overline{\lambda}_{\kappa+2}(h)z_{\kappa+2}(h) - \overline{\lambda}_{\kappa+2}(0)z_{\kappa+2}(0) \\ &+ \overline{\lambda}_{\kappa+3}(h)z_{\kappa+3}(h) - \overline{\lambda}_{\kappa+3}(0)z_{\kappa+3}(0) + \overline{\lambda}_{\kappa+4}(h)z_{\kappa+4}(h) - \overline{\lambda}_{\kappa+4}(0)z_{\kappa+4}(0) \\ &\qquad\vdots \\ &+ \overline{\lambda}_{2\kappa+1}(h)z_{2\kappa+1}(h) - \overline{\lambda}_{2\kappa+1}(0)z_{2\kappa+1}(0) + \overline{\lambda}_{2\kappa+2}(h)z_{2\kappa+2}(h) - \overline{\lambda}_{2\kappa+2}(0)z_{2\kappa+2}(0). \end{aligned}$$

However, we know that $z_j(0) = z_{j-1}(h)$ for $j = 1 \ldots \kappa, \kappa+2 \ldots 2\kappa+2$, so, we have

$$\begin{aligned}
\overline{\lambda}^T z|_0^h &= \overline{\lambda}_0(h)z_0(h) - \overline{\lambda}_0(0)z_0(0) + \overline{\lambda}_1(h)z_1(h) - \overline{\lambda}_1(0)z_0(h) \\
&+ \overline{\lambda}_2(h)z_2(h) - \overline{\lambda}_2(0)z_1(h) + \overline{\lambda}_3(h)z_3(h) - \overline{\lambda}_3(0)z_2(h) \\
&\vdots \\
&+ \overline{\lambda}_{\kappa+1}(h)z_{\kappa+1}(h) - \overline{\lambda}_{\kappa+1}(0)z_{\kappa+1}(0) + \overline{\lambda}_{\kappa+2}(h)z_{\kappa+2}(h) - \overline{\lambda}_{\kappa+2}(0)z_{\kappa+1}(h) \\
&+ \overline{\lambda}_{\kappa+2}(h)z_{\kappa+2}(h) - \overline{\lambda}_{\kappa+2}(0)z_{\kappa+2}(h) + \overline{\lambda}_{\kappa+4}(h)z_{\kappa+4}(h) - \overline{\lambda}_{\kappa+4}(0)z_{\kappa+3}(h) \\
&\vdots \\
&+ \overline{\lambda}_{2\kappa+1}(h)z_{2\kappa+1}(h) - \overline{\lambda}_{2\kappa+1}(0)z_{2\kappa}(h) + \overline{\lambda}_{2\kappa+2}(h)z_{2\kappa+2}(h) - \overline{\lambda}_{2\kappa+2}(0)z_{2\kappa+1}(h).
\end{aligned}$$

Factoring the $z's$, we get

$$\begin{aligned}
\overline{\lambda}^T z|_0^h &= -\overline{\lambda}_0(0)z_0(0) \\
&\quad +z_0(h)(\overline{\lambda}_0(h) - \overline{\lambda}_1(0)) \\
&\quad +z_1(h)(\overline{\lambda}_1(h) - \overline{\lambda}_2(0)) \\
&\quad +z_2(h)(\overline{\lambda}_2(h) - \overline{\lambda}_3(0)) \\
&\quad \vdots \\
&\quad -\overline{\lambda}_{\kappa+1}(0)z_{\kappa+1}(0) \\
&\quad +z_{\kappa+1}(h)(\overline{\lambda}_{\kappa+1}(h) - \overline{\lambda}_{\kappa+2}(0)) \\
&\quad +z_{2\kappa+1}(h)(\overline{\lambda}_{2\kappa}(h) - \overline{\lambda}_{2\kappa+1}(0)) \\
&\quad \vdots \\
&\quad +\overline{\lambda}_{2\kappa+2}(h)z_{2\kappa+2}(h).
\end{aligned}$$

Then we have

$$\begin{aligned}
J_0 - J &= (\hat{z}_0^T(0) - z_0^T(0))\beta Q_0 z_0(0) + (\hat{z}_{\kappa+1}^T(0) - z_{\kappa+1}^T(0))(1-\beta)Q_1 z_{\kappa+1}(0) - \lambda^T z|_0^h \\
&\quad + < \dot{\lambda}, \hat{z} > + < \lambda, \dot{\hat{z}} > \\
&= \hat{z}_0^T(0)\beta Q_0 z_0(0) - z_0^T(0)\beta\beta Q_0 z_0(0) + \hat{z}_{\kappa+1}^T(0)(1-\beta)Q_1 z_{\kappa+1}(0) \\
&\quad - z_{\kappa+1}^T(0)(1-\beta)(1-\beta)Q_1 z_{\kappa+1}(0) - \overline{\lambda}_0(0) z_0(0) - \overline{\lambda}_{\kappa+1}(0) z_{\kappa+1}(0) \\
&\quad + < \dot{\lambda}, \hat{z} > + < \lambda, \dot{\hat{z}} > \text{ by the boundary conditions} \\
&= \hat{z}_0^T(0)\beta Q_0 - [z_0^T(0)\beta\beta Q_0 + \overline{\lambda}_0(0)] z_0(0) + \hat{z}_{\kappa+1}^T(0)(1-\beta)Q_1 \\
&\quad - [z_{\kappa+1}^T(0)(1-\beta)(1-\beta)Q_1 + \overline{\lambda}_{\kappa+1}(0)] z_{\kappa+1}(0) + < \dot{\lambda}, \hat{z} > + < \lambda, \dot{\hat{z}} > \\
&\quad \text{by the boundary conditions} \\
&= \hat{z}_0^T(0)\beta Q_0 + \hat{z}_{\kappa+1}^T(0)(1-\beta)Q_1 + < \dot{\lambda}, \hat{z} > + < \lambda, \dot{\hat{z}} > \\
&\quad \text{by the boundary conditions} \\
&= 0 \quad \text{by going through the process above again.}
\end{aligned}$$

So, $J_0 - J = 0$ and we have proved the optimality of our solution. ◊

3.2.3 Minimum Energy Detection Signal Algorithm for the Method of Steps

Having found the necessary conditions for the inner minimum, we are able to concisely state the algorithm for finding the minimum energy detection signal for the reformulated delayed problem. The optimal control problem consists of

1. The necessary conditions for $\min_{\overline{z},\overline{\mu}} \left\{ \overline{z}(0)^T \overline{Q}_\beta \overline{z}(0) + \frac{1}{2}\int_0^h \overline{\mu}^T V_\beta \overline{\mu}\, dt \right\}$.
2. An expression with value $\min_{\overline{z},\overline{\mu}} \left\{ \overline{z}(0)^T \overline{Q}_\beta \overline{z}(0) + \frac{1}{2}\int_0^h \overline{\mu}^T V_\beta \overline{\mu}\, dt \right\}$.
3. The constraint $\min_{\overline{z},\overline{\mu}} \left\{ \overline{z}(0)^T \overline{Q}_\beta \overline{z}(0) + \frac{1}{2}\int_0^h \overline{\mu}^T V_\beta \overline{\mu}\, dt \geq 1 \right\}$.
4. The boundary conditions.

MEDS algorithm incorporating the Method of Steps

1. Find $\bar{\kappa} = \frac{\omega}{h}$.

2. Form $\overline{A}, \overline{B}, \overline{C}, \overline{M}, \overline{N}$ according to the notation of Section 3.2.1 .

3. Solve

$$\min \|\overline{v}\|^2$$

subject to

$$\begin{aligned}
\dot{\overline{z}} &= \overline{A}\overline{z} + \overline{B}\overline{v} + \overline{M}\overline{\mu} \\
\dot{\overline{\lambda}} &= -\overline{A}^T\overline{\lambda} + \overline{C}^T\eta \\
\dot{Z} &= \frac{1}{2}\overline{\mu}^T V_\beta \overline{\mu}, \\
0 &= V_\beta\overline{\mu} - \overline{M}^T\overline{\lambda} + \overline{N}^T\eta \\
0 &= \overline{C}\overline{z} + \overline{N}\overline{\mu}. \\
\overline{\lambda}_j(0) &= \overline{\lambda}_{j-1}(h) \quad \text{for} \quad j = 1\ldots\kappa, \kappa+2\ldots 2\kappa+2 \\
\overline{z}_j(0) &= \overline{z}_{j-1}(h) \quad \text{for} \quad j = 1\ldots\kappa, \kappa+2\ldots 2\kappa+2 \\
\overline{\lambda}_0(0) &= -2\beta\overline{Q}_0\overline{z}_0(0) \\
\overline{\lambda}_{\kappa+1}(0) &= -2(1-\beta)\overline{Q}_1\overline{z}_{\kappa+1}(0) \\
\overline{\lambda}_j(h) &= 0 \quad \text{for} \quad j = \kappa, 2\kappa+2 \\
Z(0) &= \overline{z}_0(0)^T\overline{Q}_\beta\overline{z}_0(0) \\
Z(h) &\geq 1 \\
& .01 \leq \beta \leq .99
\end{aligned}$$

where $\frac{1}{2}\overline{\mu}^T V_\beta\overline{\mu} = \beta\|\overline{\mu}_0\|^2 + (1-\beta)\|\overline{\mu}_1\|^2$ and $\overline{Q}_\beta = \text{diag}\{\beta\overline{Q}_0, (1-\beta)\overline{Q}_1\}$.

3.2.4 Problem Variations

Variation in the control

There are several possible problem variations that we can solve with our approach. For example, sometimes, the problem will have a control or reference trajectory, u, that acts along the same channels as the detection signal. Such a model would have the form

$$\begin{aligned}
\dot{x}_i &= A_i x_i(t) + G_i x_i(t-h) + B_i v(t) + B_i u(t) + M_i \mu_i(t), \quad t \geq 0 \\
x_i(0) &= \eta, \\
x_i(t) &= \phi_i(t) \ \text{ a.e. } \quad \text{on } \ [-h, 0) \\
y_i(t) &= C_i x_i(t) + N_i \mu_i(t), \ t \geq 0.
\end{aligned}$$

This case is easily handled. By defining $\bar{u}$ in the same way as $\bar{v}$, the system after reformulation with the Method of Steps is

$$\begin{aligned}
\dot{\overline{z}} &= \overline{A}\overline{z} + \overline{B}\overline{v} + \overline{B}\overline{u} + \overline{M}\overline{\mu} \\
0 &= \overline{C}\overline{z} + \overline{N}\overline{\mu} \\
\overline{z}_i(0) &= \overline{z}_{i-1}(h) \text{ for } \ i = 1 \ldots \kappa, \kappa + 2 \ldots 2\kappa + 2.
\end{aligned}$$

Rather than finding the smallest, proper v, the minimization problem becomes find the smallest v with proper $v + u$.

Delay in the control

A delay in the control is handled similarly to the way a delay in the state is handled. The reformulation of the problem is done in the same way. An important difference between a delay in the state and a delay in the control to note is that if there are some elements in the output set which are not immediately effected by the auxiliary signal, then there may be a minimum time, ω, below which model identification is not possible.

Other costs

Another variation on the problem would be using other possible costs. The cost we use, (1.10), is designed so that the test uses the smallest signal. However, other costs may be more appropriate.

For example, one might want a weight on the initial function, ϕ. Then we have the cost function

$$\mathcal{S}_i(x_i(0), \mu_i, \phi_i) = x_i(0)^T Q_i x_i(0) + \int_{-h}^{0} \phi_i(t)^T P \phi_i(t) \ \ dt + \int_0^{\omega} |\mu_i(t)|^2 \ \ dt < 1, \quad i = 0, 1.$$

Since ϕ is included in the noise vector μ, this modification in the noise function involves a change in the matrix, $\overline{M}$. That is, we let

$$\overline{M}_i = \begin{bmatrix} G_i P^{\frac{1}{2}} & M_i & 0 & \dots \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & M_i \end{bmatrix}.$$

Another possible variation in the cost is measuring the initial function as a difference from a trajectory, $r_i(t)$. In that case, our noise measure becomes

$$\mathcal{S}_i(x_i(0), \mu_i, \phi_i) = x_i(0)^T Q_i x_i(0) + \int_{-h}^{0} |\phi_i(t) - r_i(t)|^2 \ \ dt + \int_0^\omega |\mu_i(t)|^2 \ \ dt < 1, \quad i = 0, 1.$$

The changes in our problem set up include letting $\delta_i = \phi_i - r_i$. Then we let

$$\begin{aligned} \dot{\overline{z}} &= \overline{A}\overline{z} + \overline{B}\overline{v} + \overline{H}\overline{u} + \overline{M}\overline{\mu} \\ 0 &= \overline{C}\overline{z} + \overline{N}\overline{\mu} \\ \overline{z}_i(0) &= \overline{z}_{i-1}(h) \text{ for } \ i = 1 \dots \kappa, \kappa + 2 \dots 2\kappa + 2 \end{aligned}$$

where $\overline{\mu} = \begin{bmatrix} \delta_i \\ \mu_{i,0} \\ \vdots \end{bmatrix}$, $\overline{H}_i = \begin{bmatrix} G_i & 0 & \dots & \dots \\ 0 & 0 & \ddots & \vdots \\ & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 \end{bmatrix}$, $\overline{u} = \begin{bmatrix} r_i \\ 0 \\ \vdots \end{bmatrix}$. Then the problem can be formulated as before.

It may be that we are less interested in minimizing the disturbance to the system than we are in having the system be close to 0 at the end of the test period. Or, we may be interested in leaving some specific part of the system undisturbed while another part of the system can withstand a strong disturbance. For such scenarios as these, we might have a cost measure

$$\mathcal{S}_i(\xi(\omega), \mu_i) = \xi_i(\omega)^T Q_i \xi_i(\omega) \ \ dt + \int_0^\omega |\mu_i(t)|^2 + \xi_i^T(t) U \xi_i(t) \ \ dt < 1, \quad i = 0, 1 \tag{3.47}$$

where Q and U are positive semi-definite matrices of appropriate sizes and ξ is a function satisfying

$$\dot{\xi}_i = F\xi_i + Ev. \tag{3.48}$$

F and E are chosen to satisfy design considerations of the system being tested. For the case without delays (3.47) and (3.48) are discussed in [16].

Integral delay terms

Another possible variation in these problem is the inclusion of integral terms with delays. In this case, the model starts as

$$\dot{x}(t) = Ax(t) + Gx(t-h) + \int_{-h}^{0} Ex(t+\tau)\ \ d\tau + Bv + \int_{-h}^{0} Fv(t+\tau)\ \ d\tau + M\mu(t).$$

Note that we have a distributed delay in the state and in the control. To formulate this problem using the Method of Steps, we rewrite the delayed integrals as

$$\int_{-h}^{0} g(t+\tau)\ d\tau = \int_{t-h}^{t} g(s)\ ds = \int_{ih}^{t} g(s)\ ds + \int_{t-h}^{ih} g(s)\ ds.$$

New variables are given as additional differential equations and boundary conditions. That is, we write

$$\begin{aligned}
\dot{x}(t) &= Ax(t) + Gx(t-h) + Ex(z_i + w_{i-1}) + Bv + Fv(u_i + r_{i-1}) + M\mu(t) \\
\dot{z}_i(t) &= x_i(t), \quad z_i(0) = 0, \quad i > 0 \\
\dot{w}_i(t) &= x_i(t), \quad w_i(h) = 0, \quad i > 0 \\
\dot{u}_i(t) &= v_i(t), \quad u_i(0) = 0, \quad i > 0 \\
\dot{r}_i(t) &= v_i(t), \quad r_i(h) = 0, \quad i > 0 \\
\dot{w}_0 &= \phi, \quad w_0(h) = 0 \\
\dot{r}_0 &= \psi.
\end{aligned}$$

ϕ is given as the past history of the state and given in the normal problem statement. ψ is the past history of the detection signal. As the detection signal has no past history, a logical value for ψ is $\psi = 0$.

Chapter 4

Approximated Solutions: PDE's and Central Differences

We now move on to the more general delayed system. We assume we have models of the form

$$
\begin{aligned}
\dot{x}_i(t) &= A_i x(t) + \sum_{j=1}^{r_i} G_i x_i(t+\theta_{i,j}) + B_i v(t) + M_i \mu_i(t), \quad t \geq 0 && \text{(4.1a)}\\
x_i(0) &= x_{i0}, \;\; x_i(t) = \phi_i(t) \;\; \text{a.e. on } [-h, 0) && \text{(4.1b)}\\
y(t) &= C_i x_i(t) + N_i \mu_i(t), \; t \geq 0 && \text{(4.1c)}
\end{aligned}
$$

where $-h = \theta_l < \cdots < \theta_0 = 0$. We assume A_i, B_i, C_i, M_i, and N_i are system matrices, $v \in L^2[0, \omega]$ is an auxiliary signal, and $\mu_i \in L^2[0, \omega]$ are noises. We also continue to assume that $\omega > h$. The initial condition, x_{i0}, and the initial function, ϕ_i, are taken to be disturbances or noises. As mentioned in the previous chapter, the noise bound for this system is

$$
\mathcal{S}_i(x_i(0), \phi_i, \mu_i) = x_i(0)^T Q_i x_i(0) + \int_{-h}^{0} |\phi_i(t)|^2 \;\; dt + \int_0^{\omega} |\mu_i(t)|^2 \;\; dt \tag{4.2}
$$

where Q_i is symmetric, positive, semi-definite. Note that it is possible to have systems which are still more general. Those systems may include integral terms, as was seen in the previous chapter and as will be seen in Chapter 5.

In the previous chapter, we solved the delayed detection problem using the Method of Steps. Unfortunately, the Method of Steps, while providing a 'true' solution works only for a small class of problems. It does not apply to most problems with multiple delays.

Moreover, even when it does apply, if the delay is small relative to the detection horizon, finding the solution using the Method of Steps becomes so computationally intensive as to be impractical. On the other hand, to approach the delayed detection problem directly involves using the theory of optimal control of delayed systems which is a path fraught with technical difficulties.

In order to solve our fault detection problem, we must find the necessary conditions for the inner minimum described in (1.18). The necessary conditions for an LQR problem are not straight forward. Suppose we have the equation

$$\dot{x} + \sum_{i=0}^{N} B_i(t)x(t-h_i) + \int_{-h}^{0} B(t,\tau)x(t+\tau)\ d\tau + C(t)u(t) = 0 \qquad t \geq 0$$

$$\begin{aligned} x(0) &= \phi^0 \\ x(t) &= \phi^1(t) \qquad t \geq 0 \\ \phi &= (\phi^0, \phi^1) \in H \end{aligned}$$

where $H = \mathbf{R}^n \times L^2(-h, 0; \mathbf{R}^n)$ and we are minimizing the cost function $J(u)$ where

$$J(u) = x(\omega)^T[Qx(\omega) + 2q] + \int_0^{\omega} [x(t)^T W(t)x(t) + 2w(t)^T x(t) + u(t)^T u(t)]\ dt \tag{4.4}$$

and $Q = Q^T \geq 0$, $W(t) = W(t)^T \geq 0$ for every $t \in [0, T]$, W is bounded and absolutely integrable, and w is square integrable. Then, from [22] we take the following theorem, which gives the necessary conditions for finding the optimal control.

Theorem 3. *For every initial condition $\phi \in H$ of (4.3) in the problem (4.4), there is a unique optimal control $u \in L^2(-h, 0; \mathbf{R}^m)$. The optimal control is fully characterized by the following set of canonical equations, satisfied almost everywhere on* $[0,\ \omega]$ *[22, pp. 334]:*

$$\begin{aligned} \dot{x} &= -\sum_{i=0}^{N} B_i(t)x(t-h_i) - \int_{-h}^{0} B(t,\tau)x(t+\tau)\ d\tau + C(t)u(t) \qquad t \geq 0 && \text{(4.5a)} \\ \dot{p}(t) &= \sum_{i=0}^{N} B_i(t+h_i)p(t+h_i) + \int_{-h}^{0} B(t-\tau,\tau)^T p(t-\tau)\ d\tau && \text{(4.5b)} \\ &\quad + W(t)x(t) + w(t) \end{aligned}$$

with

$$p(\omega) = Qx(\omega) + q, \tag{4.6a}$$

$$p(t) = 0, \quad t > \omega \tag{4.6b}$$

$$u(t) = C(t)^T p(t) \tag{4.6c}$$

$$x(0) = \phi^0 \tag{4.6d}$$

$$x(t) = \phi^1(t) \quad t \geq 0 \tag{4.6e}$$

$$\phi = (\phi^0, \phi^1) \in H \tag{4.6f}$$

At first glance, these conditions might not seem to present technical difficulties. However, it must be kept in mind that in order to use these conditions, we must first reformulate our models, eliminating the output, y, in (4.1c). Then we need to rewrite our system so that this algebraic constraint is included in (4.5a) by making the coefficient matrices singular. Unfortunately, existence and uniqueness of the solutions to this problem depend on the coefficient matrix of $\dot{x}$ having a bounded inverse. In addition, the equations defining the necessary conditions are not strictly retarded. For models in which the coefficient matrices are not constant, the necessary conditions introduce advanced 'delays' in the coefficients. The necessary conditions are a hybrid of retarded and advanced functional differential equations. Developing a new theory for solving the fault detection problem with these equations is a non-trivial task.

Another way to approach these problems is to approximate the delayed system with an ODE system. Then the well-developed theory of Section 1.3.1 can be applied. Of course, in approximating, questions are raised about how accurate the approximations are, how close the detection signal of the approximated solution is to that of the 'true' detection signal, how close the approximated output set is to that of the 'true' output set, among other things. It should be noted that because of the nature of our problem, our interest is in finding a good approximation of the detection signal, rather than the optimal states.

There are many ways of approximating delayed systems with ODE's as we discussed in Section 1.2.1. Since different approximation schemes can lead to different models, it is best to use the same scheme across all the models. There is a risk that if different schemes are used, the detection signal found will be tuned to detecting differences due to the approximation scheme, rather than in the original models. In the next two chapters,

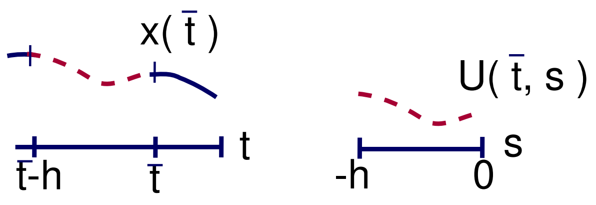

Figure 4.1: We reformulate the delayed system into a PDE by letting $U(t, s) = x(t + s)$ for $0 \leq t \leq \omega, -h \leq s \leq 0$. In the figure, $U(\bar{t}, s)$ is $x(\bar{t})$. The dashed section of the curve represents $U(\bar{t}, s)$ for $-h \leq s \leq 0$; that is, $U(\bar{t}, s)$ and its past history.

we examine two methods of approximation. In Chapter 4, we look at a reformulation of the delayed system as a PDE and an approximation of the resulting PDE with central differences. In Chapter 5, we look at a spline approximation from Ito and Kappel's *A Uniformly Differentiable Approximation Scheme for Delay Systems Using Splines*[34].

4.1 Reformulating the Delayed System with a PDE

First, we reformulate the delayed system with a PDE. Suppose that we have one of the models of the form (4.1). We suppress the subscript i to simplify the notation and describe the approximation process for a single model. Let $U(t, s) = x(t + s)$ for $0 \leq t \leq \omega, -h \leq s \leq 0$. Then for a fixed $\bar{t}$, $U(\bar{t}, s)$ is a function in L^2 which is x on the interval $[\bar{t} - h,\ \bar{t}]$. That is, for a fixed $\bar{t}$, $U(\bar{t}, s)$ is the value of $x(\bar{t})$ and its history. Figure 4.1 illustrates this.

In order to ensure we have the right model, we define the following partial differential equation:

$$\begin{aligned}
U_t(t,s) &= U_s(t,s) && (4.7a)\\
U_t(t,0) &= AU(t,0) + \sum_{j=1}^{r} G_j U(t,-\theta_j) + Bv(t) + M\mu(t) && (4.7b)\\
U(0,s) &= \phi(s) && (4.7c)\\
U(0,0) &= x_0. && (4.7d)
\end{aligned}$$

4.2 Explanation of the Central Differences Approximation

There are many ways of approximating PDE's such as (4.7). One method is the use of central differences and the method of lines. Again, we temporarily suppress the i subscript. We choose a mesh for $[-h, 0]$ of the form

$$-h = s_0 < s_1 < \cdots s_\rho = 0. \tag{4.8}$$

The one restriction on this mesh is that each $-\theta_j$ has to be a mesh point s_{m_j}. We suppose then that $-\theta_j = s_{m_j}$. The value of h and the delays may vary from model to model. Let

$$U_k(t) = U(t, s_k).$$

Then we get the following ODE system

$$\begin{aligned}
U_0'(t) &= \sum_{h=0}^{2} \alpha_{0,h} U_h(t) && (4.9a)\\
U_k'(t) &= \sum_{h=-1}^{1} \alpha_{k,h} U_{k+h}(t), \quad 0 < k < \rho - 1 && (4.9b)\\
U_\rho'(t) &= AU_\rho(t) + \sum_{j=1}^{r} G_j U_{m_j}(t) + Bv(t) + M\nu(t) && (4.9c)\\
U_k(0) &= \phi(s_k) && (4.9d)\\
U_\rho(0) &= x_0 \quad (\text{ or } \phi_i(0)) && (4.9e)
\end{aligned}$$

where the sum in (4.9c) is an approximation for the spatial derivative. If we have $0 < k < \rho$

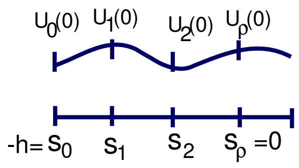

Figure 4.2: Having reformulated the problem into a PDE, we now use the method of lines and central differences to approximate the PDE with an ODE. We first define a mesh $-h = s_0 < s_1 < \cdots s_\rho = 0$, where each delay must be a mesh point. Then we let $U_k(t) = U(t, s_k)$. This illustration represents the new variables when $t = 0$. Notice that the new variables will represent the past history needed because of the restriction that each delay must be a mesh point.

with $\delta = s_{k+1} - s_k$ and $\epsilon = s_k - s_{k-1}$, then we can take

$$\alpha_{k,1} = \frac{\epsilon}{\delta(\epsilon+\delta)}, \qquad \alpha_{k,-1} = -\frac{\delta}{\epsilon(\epsilon+\delta)}, \qquad \alpha_{k,0} = \frac{\epsilon}{\delta(\epsilon+\delta)} - \frac{\delta}{\epsilon(\epsilon+\delta)}. \tag{4.10}$$

This provides an approximation for the spatial derivative in (4.9c) which is $O(\delta\epsilon)$. To get a similar accuracy for U_0 in equation (4.9a) we need to use a one sided approximation using two extra values. Let $\delta = s_1 - s_0,\ \epsilon = s_2 - s_1$. We can then take

$$\alpha_{0,0} = -\alpha_{0,2} - \alpha_{0,1}, \qquad \alpha_{0,1} = \frac{\delta+\epsilon}{\delta\epsilon}, \qquad \alpha_{0,2} = -\frac{\delta}{\epsilon(\delta+\epsilon)}.$$

Two things should be noticed. First we have

$$a_{k,1} \le \frac{1}{\delta}, \qquad a_{k,-1} \le \frac{1}{\epsilon}$$

so that the coefficients only grow as one over the mesh size. Secondly if $\epsilon = \delta$, then (4.10) simplifies to

$$a_{k,1} = \frac{1}{2\delta}, \qquad a_{k,0} = 0, \qquad a_{k,-1} = -\frac{1}{2\delta}.$$

This has transformed the delay model equation to an approximate ordinary differential equation model. We also must transform the noise measure (4.2). Let $U_{i,j}$ be the j^{th} vector in the approximation for model i. Let $\gamma_{i,j}$ be the collocation coefficients for approximating integrals on $[-h_i, 0]$ using the grid points $s_{i,j}$. Then we can approximate $\int_{-h}^{0} |\phi_i(t)|^2 \ dt$

with $\sum_{j=0}^{\rho_i-1} \gamma_{i,j}\|U_{i,j}(0)\|^2 + \gamma_{i,\rho_i}|\phi_i(0)|^2$. Notice that we do not say that $U_{i,j}(0) = \phi_i(0)$ as $U_{i,j}(0) = x_i(0)$ and we have no assumption of continuity between our initial function and initial condition. Then our noise measure, (4.2), becomes

$$\begin{aligned}\widetilde{\mathcal{S}}_i(\phi_i, \mu_i, x_{i,0}) &= \sum_{j=0}^{\rho_i-1} \gamma_{i,j}\|U_{i,j}(0)\|^2 + \gamma_{i,\rho_i}|\phi_i(0)|^2 &\text{(4.14a)}\\ &+U_{i,\rho_i}(0)^T Q_i U_{i,\rho_i}(0) + \int_0^{\omega} \|\mu_i\|^2\, dt \le 1. &\text{(4.14b)}\end{aligned}$$

We use collocation schemes for which all the $\gamma_{i,j} > 0$. For example, if we are using a Trapezoidal approximation, then we have (suppressing the i subscript on all terms)

$$\gamma_0 = \frac{s_1 - s_0}{2}, \quad \gamma_\rho = \frac{s_\rho - s_{\rho-1}}{2}, \quad \gamma_j = \frac{s_{j+1} - s_{j-1}}{2} \text{ if } j \notin \{0, \rho\}. \tag{4.15a}$$

4.3 Necessary Conditions for the Fault Detection Problem

Now that we have transformed the delayed system to an ODE system, we can explicitly state the optimization problem which needs to be solved. First, we define some notation. While the notation we will define is for the case of two models, it easily and obviously extends to more than two models. We let

$$\bar{r} = \begin{bmatrix} r_0 \\ r_1 \end{bmatrix}$$

for $\bar{r} = U, \bar{\mu}$ where $U_i = \begin{bmatrix} U_{i,0} \\ \vdots \\ U_{i,\rho} \end{bmatrix}$. We let

$$\bar{R} = \begin{bmatrix} \bar{R}_0 & 0 \\ 0 & \bar{R}_1 \end{bmatrix}$$

for $\bar{R} = \bar{A}, \bar{B}, \bar{M}$ where

$$\bar{A}_i = \begin{bmatrix} \alpha_{0,0} & \alpha_{0,1} & \alpha_{0,2} & 0 & \cdots & \cdots & \cdots & 0 \\ \alpha_{1,-1} & \alpha_{1,0} & \alpha_{1,1} & 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \alpha_{2,-1} & \alpha_{2,0} & \alpha_{2,1} & 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & \alpha_{k,-1} & \alpha_{k,0} & \alpha_{k,1} & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 & \alpha_{\rho-1,-1} & \alpha_{\rho-1,0} & \alpha_{\rho-1,1} \\ G_{i,\rho} & \cdots & \cdots & \cdots & \cdots & G_{i,2} & G_{i,1} & A_i \end{bmatrix}$$

with $G_{i,k}$ defined in (4.7) or 0 otherwise, $\bar{B}_i = \begin{bmatrix} 0 \\ B_i \end{bmatrix}$, and $\bar{M}_i = \begin{bmatrix} 0 \\ M_i \end{bmatrix}$. We let

$$\bar{P} = \begin{bmatrix} \bar{P}_0 & 0 \\ 0 & -\bar{P}_1 \end{bmatrix}$$

for $\bar{P} = \bar{N}, \bar{C}$ where $\bar{C}_i = \begin{bmatrix} 0 & C_i \end{bmatrix}$ and $\bar{N}_i = \begin{bmatrix} 0 \\ N_i \end{bmatrix}$. We let $\bar{Q}_i = \text{diag}\{\gamma_{i,0} \ \cdots \ \gamma_{i,\rho-1} \ Q_i\}$, so that $\frac{1}{2}\bar{Q}_\beta = \begin{bmatrix} \beta\bar{Q}_0 & 0 \\ 0 & (1-\beta)\bar{Q}_1 \end{bmatrix}$, and $\frac{1}{2}\bar{\mu}^T V_\beta \bar{\mu} = \beta\|\bar{\mu}_0\|^2 + (1-\beta)\|\bar{\mu}_1\|^2$.

Then in this notation, the system can be written as

$$\begin{aligned} \dot{U} &= \bar{A}U + \bar{B}v + \bar{M}\bar{\mu} \\ 0 &= \bar{C}U + \bar{N}\bar{\mu} \end{aligned}$$

and the noise measure can be written as

$$\bar{\mathcal{S}}(U(0), \bar{\mu}) = \frac{1}{2}U(0)^T \bar{Q}_\beta U(0) + \sum_{i=0}^{1} \gamma_{i,\rho_i} |\phi_i(0)|^2 + \frac{1}{2}\bar{\mu}^T V_\beta \bar{\mu} \ \ dt.$$

Thus, finding necessary conditions for the inner minimum means we must find

$$\min_{U,\mu} \quad \left\{ \frac{1}{2}U(0)^T \bar{Q}_\beta U(0) + \sum_{i=0}^{2} \gamma_{i,\rho_i} |\phi_i(0)|^2 + \frac{1}{2}\bar{\mu}^T V_\beta \bar{\mu} \ \ dt \right\} \tag{4.19a}$$

$$\text{subject to} \quad \dot{U} = \bar{A}U + \bar{B}v + \bar{M}\bar{\mu} \tag{4.19b}$$

$$0 = \bar{C}U + \bar{N}\bar{\mu}. \tag{4.19c}$$

First we note that $\phi_i(0)$ appears only in the noise measure (4.19a) and is completely independent of the states and the control. Thus, it appears only in the cost of our optimization problem. Since we have a positive term appearing only in the cost, we see that we need the extra condition that $\phi_i(0) = 0$. Failure to use the extra condition, $\phi_i(0) = 0$, while using the rest of the approximation leads to ridiculous approximations of the detection signal, v. As we saw while working our examples, not only were the signals found wrong, they were near 0. Also, neglecting to use the condition, $\phi_i(0) = 0$, causes problems for the optimization software.

Using the theory developed in Section 1.3.1, we have that

$$\begin{aligned}
\dot{U} &= \overline{A}U + \overline{B}v + \overline{M}\mu \\
\dot{\lambda} &= -\overline{A}^T\lambda + \overline{C}^T\eta. \\
0 &= V_\beta\mu - \overline{M}^T\lambda + \overline{N}^T\eta \\
0 &= \overline{C}U + \overline{N}\mu. \\
\lambda(0) &= -\overline{Q}_\beta U(0) \\
\lambda(\omega) &= 0.
\end{aligned}$$

The optimal control problem using differences

Having found the necessary conditions for the inner minimum, we can now state the optimal control problem which must be solved. We must find

$$\min \|\overline{v}\|^2$$

subject to

$$
\begin{aligned}
\dot{U} &= \overline{A}U + \overline{B}v + \overline{M}\mu && (4.21a)\\
\dot{\lambda} &= -\overline{A}^T\lambda + \overline{C}^T\eta && (4.21b)\\
\dot{Z} &= \frac{1}{2}\mu^T V_\beta \mu, && (4.21c)\\
0 &= V_\beta\mu - \overline{M}^T\lambda + \overline{N}^T\eta && (4.21d)\\
0 &= \overline{C}U + \overline{N}\mu && (4.21e)\\
\lambda(0) &= -\overline{Q}_\beta U(0) && (4.21f)\\
\lambda(\omega) &= 0 &&\\
Z(0) &= U(0)^T\overline{Q}_\beta U(0) && (4.21g)\\
Z(\omega) &\geq 1 && (4.21h)\\
\phi_i(0) &= 0 && (4.21i)\\
.01 \leq \beta &\leq .99. && (4.21j)
\end{aligned}
$$

Also, it should be clear that as the size of the mesh we choose in (4.8) grows, so does the size of our optimization problem. As we have $\sum_{i=0}^{n}\dim(x_i)$ states from the n models, when we approximate the systems, we have $\sum_{i=0}^{n}\rho_i\dim(x_i)$ states. Then finding the necessary conditions adds extra states to the optimization problem, so that we have $\sum_{i=0}^{n}[2\rho_i\dim(x_i)]$ states. The cost (4.21c) adds another state to the problem. One can see that the number of states in our optimization problem becomes $\sum_{i=0}^{n}[2\rho_i\dim(x_i)]+1$. Thus, increasing the size of ρ_i can increase the size of the problem significantly. Because of this, when working with these approximations, it will be in our desire to work on as coarse a grid as possible.

In Chapter 6, we will see that the difference approximation does a good job of approximating the detection signal. However, first we turn our attention to another type of approximation.

Chapter 5

Approximated Solutions: Spline Approximations

Next, we look at a spline approximation scheme adapted from Ito and Kappel's paper *A Uniformly Differentiable Approximation Scheme for Delay Systems Using Splines* [34]. A difference between our system and Ito and Kappel's system is our noise term. However, as our noises are also considered controls, if we consider that $B = \begin{bmatrix} B & M \end{bmatrix}$ and $u = \begin{bmatrix} v \\ \mu \end{bmatrix}$, then our system is in the same form as Ito and Kappel's. We will approach this problem in the same way that we have approached the problem using the Method of Steps and differences. That is, we will reformulate the problem using the spline method and then apply the theory developed in Section 1.2.

First, we offer some of the details of the spline approximation given in [34]. Then we discuss the necessary conditions and formulation of the optimal control problem. Then we discuss some theoretical results concerning the accuracy of the approximation of the detection signal, v. In particular, we explain the good approximations for v which are found though the approximations do not converge uniformly in ϕ.

5.1 Explanation of the Spline Approximation

In this subsection, we detail our adaptation of the spline approximation developed by Ito and Kappel in [34]. Our explanation differs from Ito and Kappel's in that it includes the noise term μ as discussed above. We have the delay equation

$$\begin{aligned}\dot{x}_i(t) &= \sum_{j=1}^{r_i} A_i x(\theta_{i,j}) + \int_{-h}^{0} \tilde{A}_i(s)x(t+s)\ ds + B_i v(t) + M_i \mu_i(t), \quad t \geq 0 \qquad (5.1a)\\ x_i(0) &= x_{i0}, \quad x_i(t) = \phi(t) \ \text{ a.e. on } \ [-h,0) \qquad (5.1b)\\ y(t) &= C_i x_i(t) + N_i \mu_i(t), \ t \geq 0. \qquad (5.1c)\end{aligned}$$

where $-h = \theta_l < \cdots < \theta_0 = 0$, $x(t) \in \mathbf{R}^n$, $v(t) \in \mathbf{R}^m$, $y(t) \in \mathbf{R}^p$, $\mu(t) \in \mathbf{R}^l$, $\tilde{A}_i(\cdot)$ is $n \times n$ in $L^2[-h,0]$ and A_i, B_i, C_i, M_i, N_i, G_i, $\tilde{A}_i(s)$ are matrices, v is an auxiliary signal, and the μ_i are noises.

Reformulating the problem in terms of the semi-group S

For simplicity, we drop the subscript on the models in this part of the discussion. Let $S(t)$, for $t > 0$ be

$$S(t)(\eta,\phi) = (x(t), x_t), \ t \geq 0, (\eta,\phi) \in Z = \mathbf{R}^n \times L^2(-r,0;\mathbf{R}^n)$$

where $x(t)$ is the solution to (5.1) with $v \equiv 0$, $\mu \equiv 0$, (η,ϕ) the initial data, and x_t is $x(\theta) = x(t+\theta), \quad \theta \in [-r,0)$.

Notice that $z = (\eta,\phi) \in Z$ is a point and its past history. The family $S(\cdot)$ is a strongly continuous semi-group on $Z = \mathbf{R}^n \times L^2(-r,0;\mathbf{R}^n)$ with infinitesimal generator, $\mathcal{A}$, where $\mathcal{A}$ is given by

$$\begin{aligned} dom\mathcal{A} &= \{(\eta,\phi) \in Z | \phi \in H^1(-r,0;\mathbf{R}^n), \ \eta = \phi(0)\}\\ \mathcal{A}(\phi(0),\phi) &= (L\phi, \dot{\phi}), \ \text{ for } \ (\phi(0),\phi) \in dom\mathcal{A}\end{aligned}$$

where for continuous ϕ

$$L(\phi) = \sum_{j=0}^{l} \mathcal{A}\phi(\theta) + \int_{-r}^{0} \tilde{A}_i(s)x(t+s)ds.$$

Let $\mathcal{B} : \mathbf{R}^n \to Z$, $\mathcal{C} : Z \to \mathbf{R}^n$, $\mathcal{M} : \mathbf{R}^l \to Z$, and $\mathcal{N} : \mathbf{R}^l \to \mathbf{R}^n$ by

$$\begin{aligned}\mathcal{B}v &= (Bv, 0) \quad u \in \mathbf{R}^m\\ \mathcal{M}\mu &= (M\mu, 0) \quad \mu \in \mathbf{R}^l\\ \mathcal{N}\mu &= N\mu \quad \mu \in \mathbf{R}^l\\ \mathcal{C}(\eta,\phi) &= C\eta \quad (\eta,\phi) \in Z.\end{aligned}$$

Then our original system is equivalent to the abstract system in Z,

$$\begin{aligned} \dot{z} &= \mathcal{A}z(t) + \mathcal{B}v(t) + \mathcal{M}\mu(t) \quad t \geq 0, \quad z(0) = (\eta, \phi) && (5.4a) \\ y(t) &= \mathcal{C}z(t) + \mathcal{N}\mu(t). && (5.4b) \end{aligned}$$

So, $x : [0, \infty) \to \mathbf{R}^n$ is a solutions of our original system iff the function

$$z(t) = (x(t), x_t), \quad t \geq 0$$

is a *mild solution* of the abstract system. That is,

$$z(t) = S(t)(\eta, \phi) + \int_0^T S(t-s)(\mathcal{B}u(s) + \mathcal{M}\mu(s))ds, \quad t \geq 0.$$

The *adjoint semigroup* $S^*(\cdot)$ has infinitesimal generator $\mathcal{A}^*$ is given by

$$\begin{aligned} dom\mathcal{A}^* &= \{(y, \psi) \in Z | w \in H^1(-r, 0; \mathbf{R}^n), \ \psi(-r) = A_l^T\} \\ \mathcal{A}^*(y, \psi) &= (\psi(0) + A_0^T y, A^T(\cdot)y - \dot{w}), \quad \text{for} \ (y, \psi) \in dom\mathcal{A}^* \end{aligned}$$

where $w = \psi + \sum_{i=1}^{l-1} A_l^T y \chi_{[-r, \theta_i)}$, and χ_M denotes the characteristic function of the set M. Let $l : dom\mathcal{A} \to H^1$ be given by

$$l(\phi(0), \phi) = \phi \quad \text{for} \ (\phi(0), \phi) \in dom\mathcal{A}$$

$$l^{-1}\phi = (\phi(0), \phi) \quad \text{for} \ \phi \in H^1.$$

Then l is an isomorphism from $dom\mathcal{A}$ to H^1 with the graph norm.

The scheme

Let $t_k^N = \frac{-kr}{N}$, $k = 0, \ldots, N$, with $t_{-1}^N = 0$, $t_{N+1}^N = 0$ for $N = 1, 2, \ldots$. Let $B_k^N, k = 0, \ldots, N$ be the first order splines on the interval from $[-r, 0]$ on mesh $t_0^N, \ldots, t_N^N$. So

$$B_k^N(\theta) = \begin{cases} \frac{N}{r}(\theta - t_{k+1}^N) & \text{for} \ t_{k+1}^N \leq \theta \leq t_k^N \\ \frac{N}{r}(t_{k-1}^N - \theta) & \text{for} \ t_k^N \leq \theta \leq t_{k-1}^N \\ 0 & \text{elsewhere.} \end{cases}$$

Let

$$E_k^N(\theta) = \chi_{[t_k^N, t_{k-1}^N)} = \begin{cases} 1 & \text{for} \ [t_k^N, t_{k-1}^N) \\ 0 & \text{elsewhere.} \end{cases}$$

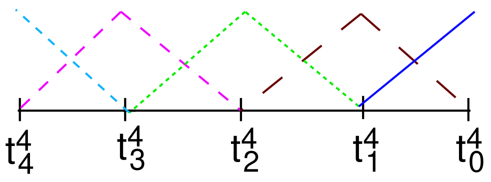

Figure 5.1: The illustration represents piecewise, linear spline functions, $B^N_k(\theta)$, of the scheme for $N = 4$. Notice that $t^N_k = \frac{-kr}{N}, k = 0, \ldots, N$ so that as k increases, t^N_k travels backward in time. Notice also that $t^N_0 = 0$ and $t^N_N = -r$ where r is the length of the longest delay. Thus, the spline functions are defined on an interval the size of the longest delay.

Let

$$\hat{E}^N_0 = (I_n, 0), \quad \hat{E}^N_k = (0, E^N_k I_n), \quad k = 1, \ldots, N.$$

We also use the spaces:

$$\begin{aligned} W^N &= span(E^N_1 I_n, \ldots, E^N_N I_n) \subset L^2(-r, 0; \mathbf{R}^n) \\ Z^N &= \mathbf{R}^n \times W^N = span(\hat{E}^N_0, \ldots, \hat{E}^N_N) \subset Z \\ X^N &= span(B^N_0 I_n, \ldots, B^N_N I_n) \subset H^1 \\ Z^N_1 &= l^{-1} X^N \subset dom\mathcal{A}. \end{aligned}$$

We have the basis matrices:

$$\begin{aligned} E^N &= (E^N_1 I_n, \ldots, E^N_N I_n) \\ \hat{E}^N &= (\hat{E}^N_0, \ldots, \hat{E}^N_N) \\ B^N &= (B^N_0 I_n, \ldots, B^N_N I_n) \\ l^{-1}B^N &= (l^{-1}B^N_0 I_n, \ldots, l^{-1}B^N_N I_n). \end{aligned}$$

Then in the notation above, $z = (\eta, \phi) \in Z^N$ can be written as

$$z^N = (\eta, E^N a^n) = \hat{E}^N \begin{bmatrix} \eta \\ a_1 \\ \vdots \\ a_N \end{bmatrix}$$

where $a_k^N \in \mathbf{R}^n$ and a^N is the coordinate vector of $\phi \in W^N$ with respect to the basis E^N.

This leads to Lemma 2.1 in [34] and the orthogonal projections $P^N : Z \to Z^N$.

Lemma 2. *In the notation and assumptions defined above, we have*

(a) For $(\eta, \phi) \in Z$

$$P^N z = (\eta, E^N a^N), \quad \textit{where} \;\; a_k^N = \frac{N}{r} \int_{t_k^N}^{t_{k+1}^N} \phi(s)\; ds, \;\; k = 1, \ldots, N.$$

(b) For $\psi \in H^1$

$$P_1^N \psi = B^N b^n, \quad \textit{where} \;\; b_k^N = \psi(t_k^N), \quad k = 0, \ldots, N.$$

Then by definition of the spaces Z_1^N, Z^n and the operators $\mathcal{A}, \mathcal{B}$, and $\mathcal{M}$, we have [34]

$$\mathcal{A}z^N \in Z^N \qquad \text{for any} \;\; z^N \in Z_1^N \tag{5.8a}$$

$$\mathcal{B}\xi \in Z^N \qquad \text{for any} \;\; \xi \in \mathbf{R}^n \tag{5.8b}$$

$$\mathcal{M}\mu \in Z^N \qquad \text{for any} \;\; \mu \in \mathbf{R}^l. \tag{5.8c}$$

To get a mild solution, $z(t)$, of the abstract system, (5.4), we need an approximation $w^N(t) \in Z_1^N$ of $z(t)$. If $z(t)$ is a strong solution of the system, then $\dot{z}$ is probably not in $dom\mathcal{A}$ but is in the subspace generated by $\mathcal{A}z(t) + \mathcal{B}v(t) + \mathcal{M}\mu(t)$. But, we know that $\mathcal{A}w^N(t) + \mathcal{B}v(t) + \mathcal{M}\mu(t) \in Z^N,\; t \geq 0$. But, $\dot{w}^N$ is in $Z_1^N \subset dom\mathcal{A}$. So, we must determine $w^N(t)$ such that

$$P^N \dot{w}^N = \frac{d}{dt} P^N w^N(t) = \mathcal{A}w^N(t) + \mathcal{B}v(t) + \mathcal{M}\mu(t), \quad t \geq 0.$$

This brings us to a Lemma 2.2 in [34](adapted to include μ).

Lemma 3. *We have*

(a) P^N *restricted to* Z_1^N *is a bijection* $Z_1^N \to Z^N$*. Its matrix representation (with respect to the basis* $l^{-1}B^N$ *of* Z_1^N *and the basis* $\hat{E}^N$ *of* Z^N *is given by*

$$Q^N = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \frac{1}{2} & \frac{1}{2} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \frac{1}{2} & \cdots & 0 & \frac{1}{2} \end{bmatrix} \otimes I_n \in \mathbf{R}^{n(N+1) \times n(N+1)}$$

(b) $\mathcal{A}$ restricted to Z_1^N is a map $Z_1^N \to Z^N$ with the matrix representation

$$\begin{aligned} H^N &= \begin{bmatrix} D_0^N & \cdots & \cdots & D_N^N \\ 0 & \cdots & \cdots & 0 \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix} \\ &+ \frac{N}{r} \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 1 & -1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -1 \end{bmatrix} \otimes I_n \in \mathbf{R}^{n(N+1)\times n(N+1)} \end{aligned}$$

where $D_k^N = L(B_K^N) = \sum_{j=0}^{l} A_j B_K^N(\theta_j) + \int_{-r}^{0} A(\theta) B_K^N(\theta)\, d\theta,\ k = 0, \ldots, N.$

(c) The matrix representation of $\mathcal{B}$ considered as a map into Z^N is

$$B^N = \begin{bmatrix} B \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbf{R}^{n(N+1)\times m}.$$

The matrix representation of $\mathcal{M}$ considered as a map into Z^N is

$$M^N = \begin{bmatrix} M \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbf{R}^{n(N+1)\times l}.$$

The matrix representation of $\mathcal{C}$ restricted to Z^N is

$$C^N = \begin{bmatrix} B & 0 & \cdots & 0 \end{bmatrix} \in \mathbf{R}^{p\times n(N+1)}.$$

The matrix representation of $\mathcal{N}$ is

$$N^N = N \in \mathbf{R}^{p\times l}.$$

Let

$$L^N = (P^N|_{Z_1^N})^{-1}, \;\; \mathcal{A}^N = \mathcal{A}L^N = \mathcal{A}(P^N|_{Z_1^N})^{-1}$$

and put

$$z^N(t) = P^N w^N(t), \qquad t \geq 0.$$

Then,

$$\dot{z}^N(t) = \mathcal{A}^N z^N(t) + \mathcal{B}v(t) + \mathcal{M}\mu(t), \qquad t \geq 0$$

and $\mathcal{A}^N$ is an operator $Z^N \to Z^N$. A^N has the matrix representation

$$A^N = H^N(Q^N)^{-1}.$$

Let $b^N(t)$ and $a^N(t)$ be the coordinate vectors of $w^N(t)$ and $z^N(t)$. That is, $z^N(t) = \hat{E}^N(\eta, a^N) = \hat{E}^N a^N(t)$. Then

$$z^N(t) = Q^N b^N(t), \qquad t \geq 0$$

and, we have

$$Q^n \dot{b}(t) = H^N b^N(t) + B^N v(t) + M^N \mu(t), \qquad t \geq 0$$

and

$$\dot{a}(t) = A^N a^N(t) + B^N v(t) + M^N \mu(t), \qquad t \geq 0.$$

Thus, we have approximated the delayed system and we have the system (including the subscripts again)

$$\dot{a}_i(t) \;\; = \;\; A_i^N a_i^N(t) + B_i^N v(t) + M_i^N \mu_i(t) \tag{5.10a}$$

$$y(t) \;\; = \;\; N_i^N a_i^N(t) + C_i^N \mu_i(t). \tag{5.10b}$$

Writing the noise measure in the new variables

Now, we must write the noise measure

$$\mathcal{S}_i(x_i(0), \mu_i) = x_i(0)^T Q_i x_i(0) + \int_{-h}^{0} |\phi_i(t)|^2 \;\; dt + \int_0^{\omega} |\mu_i(t)|^2 \;\; dt$$

in the variables of our approximation. Recall that $z = (\eta, \phi) \in Z$ is written as

$$z^N(t) = (\eta, E^N a^N) = \hat{E}^N \begin{bmatrix} \eta \\ a_1 \\ \vdots \\ a_N \end{bmatrix}.$$

Then we have

$$\begin{aligned} \mathcal{S}_i(z_i^N(0), \mu_i) &= z_{i,0}^N(0)^T Q_i z_{i,0}^N(0) + \sum_{k=1}^{N} \int_{t_k^N}^{t_{k-1}^N} |z_{i,k}^N(0) E_k^N|^2 \;\; dt + \int_0^\omega |\mu_i(t)|^2 \;\; dt \\ &= z_{i,0}^N(0)^T Q_i z_{i,0}^N(0) + \sum_{k=1}^{N} \int_{t_k^N}^{t_{k-1}^N} |z_{i,k}^N(0)|^2 \;\; dt + \int_0^\omega |\mu_i(t)|^2 \;\; dt \\ &= z_{i,0}^N(0)^T Q_i z_{i,0}^N(0) + \sum_{k=1}^{N} \frac{r_i}{N} |z_{i,k}^N(0)|^2 \;\; dt + \int_0^\omega |\mu_i(t)|^2 \;\; dt. \end{aligned}$$

If we let $Q_i^N = \text{diag}\{Q_i \; \frac{r_i}{N} \dots \frac{r_i}{N}\}$, then we can rewrite our noise measure as

$$\mathcal{S}_i(z_i^N(0), \mu_i) \;\; = \;\; z_i^N(0)^T Q_i^N z_i^N(0) + \int_0^\omega |\mu_i(t)|^2 \;\; dt.$$

5.2 The Optimal Control Problem Using Splines

Having approximated models we can now turn our attention to stating the optimal control problem which we need to solve to find the detection signal, v. Note that to avoid a conflict in notation later in this document, we are going to make a slight change in notation and assume that our approximated model i is

$$\dot{z}_i^N(t) \;\; = \;\; A_i^N z_i^N(t) + B_i^N v(t) + M_i^N \mu_i(t) \tag{5.13a}$$

$$y(t) \;\; = \;\; N_i^N z_i^N(t) + C_i^N \mu_i(t) \tag{5.13b}$$

and noise measure i is

$$\mathcal{S}_i(z_i^N(0), \mu_i) \;\; = \;\; z_i^N(0)^T Q_i z_i^N(0) + \int_0^\omega |\mu_i(t)|^2 \;\; dt.$$

We apply the theory of Section 1.2. First, we rewrite the system so that both of the models are included in a single set of matrices. Let $z^N = \begin{bmatrix} z_0^N \\ z_1^N \end{bmatrix}$. We let $R^N = \begin{bmatrix} R_0^N & 0 \\ 0 & R_1^N \end{bmatrix}$

for $R^N = A^N,\ B^N,\ M^N$. We let $P^N = \begin{bmatrix} P_0^N & 0 \\ 0 & -P_1^N \end{bmatrix}$ for $P^N = N^N,\ C^N$. Let $\frac{1}{2}Q_\beta^N = \begin{bmatrix} \beta Q_0^N & 0 \\ 0 & (1-\beta)Q_1^N \end{bmatrix}$, and $\frac{1}{2}\mu^T V_\beta \mu = \beta\|\mu_0\|^2 + (1-\beta)\|\mu_1\|^2$. Then our system and noise measure can be written as

$$\begin{aligned} \dot{z}^N(t) &= A^N z^N(t) + B^N v(t) + M^N \mu(t) && \text{(5.15a)} \\ y(t) &= N^N z^N(t) + C^N \mu(t) && \text{(5.15b)} \end{aligned}$$

and

$$\mathcal{S}(z^N(0), \mu) = \frac{1}{2} z^N(0)^T Q_\beta z^N(0) + \frac{1}{2}\mu(t)^T V_\beta \mu(t).$$

We apply the necessary conditions of Section 1.2 and we find that we need to solve the following problem:

$$\min \|v\|^2$$

$$\begin{aligned} \text{subject to} \quad \dot{z} &= A^N z^N + B^N v + M^N \mu \\ \dot{\lambda} &= -A^{NT}\lambda + C^{NT}\eta. \\ \dot{Z} &= \frac{1}{2}\mu^T V_\beta \mu \\ 0 &= V_\beta \mu - M^{NT}\lambda + N^{NT}\eta \\ 0 &= C^N z^N + N^N \mu \\ Z(0) &= \frac{1}{2} z^N(0)^T Q_\beta^N z^N(0), \quad Z(\omega) \geq 1 \\ \lambda(0) &= -Q_\beta^N z^N(0), \quad \lambda(\omega) = 0 \\ & .01 < \beta < .99. \end{aligned}$$

Solving this problem will yield the detection signal we need. As we shall see, the detection signal resulting from this approximation is excellent. Next, we turn our attention to a more theoretical look at this approximation.

5.3 Theoretical Results

In this section we examine the spline approximation more theoretically. It is important to note that our interest is not in the accuracy of the approximation of the states so much as in the accuracy of the approximation of the auxiliary signal, v. We are also interested in the approximation of the output sets, $\mathcal{A}_i$.

Suppose we have the system

$$\begin{aligned} \dot{x} &= Ax(t) + Gx(t-r) + Bu(t) + M\mu(t), \quad t \geq 0 \\ x(0) &= \eta, \quad x(t) = \phi(t) \;\text{ a.e. on } [-r, 0) \\ y(t) &= Cx(t) + N\mu(t), \; t \geq 0 \end{aligned}$$

and its approximated counterpart, $z^N = \hat{E}^N a^N$ where a^N is the solution to

$$\begin{aligned} \dot{z}^N(t) &= A^N z^N(t) + B^N v(t) + M^N \mu(t) && (5.19a) \\ y(t) &= N^N z^N(t) + C^N \mu(t) && (5.19b) \end{aligned}$$

as described in Section 5.1.

Note that in the approximation, we are dealing with state variables $z(t)^N = (x^N(t), a^N)$ where a^N is a vector of spline coefficients of the past history of $x^N(t)$. The matrices C^N, B^N, and M^N have zeros appended to them such that the dimensions for the matrix arithmetic are correct. If we are picking off the $x^N(t)$, which has the dimensions of $x(t)$, then it is also valid to use the matrices C, B, and M; and, the dimensions for the matrix arithmetic are also correct.

We assume that N has the form $[\tilde{N} \;\; 0]$ and M has the form $[0 \;\; \tilde{M}]$. Thus, we may say that $\mu = [\mu_1 \; \mu_2]$ where μ_1 and μ_2 have appropriate dimensions, so that $N\mu = \tilde{N}\mu_1$ and $M\mu = \tilde{M}\mu_2$. Then we can rewrite (5.18a) as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Gx(t-r) + Bv(t) + \tilde{M}\mu_2(t), \quad t \geq 0 && (5.20a) \\ y(t) &= Cx(t) + \tilde{N}\mu_1(t), \; t \geq 0 && (5.20b) \\ x(0) &= \eta, \quad x(t) = \phi(t) \;\text{ a.e. on } [-r, 0). && (5.20c) \end{aligned}$$

This is not too restrictive an assumption, as we are simply assuming that the noises in the dynamics and in the output are independent.

First, in Section 5.3.1, we review some well-known definitions and theorems which will be of use in our arguments. Of particular importance will be the definition of a compact linear operator and the Compactness Criterion. While the spaces we are working in are not compact, the fact that our operators are compact will allow us to use compactness techniques in our proofs.

In Section 5.3.2, we prove a few lemmas. It is hoped that the use of the lemmas will make the proofs of our main results more transparent to the reader. Lemma 8 is especially important. In this lemma we find a finite number of elements in the permissible initial data set so that image of every element in the set is close to the image of at least one of those elements.

In the subsequent sections are our main results. In our main results, we try to answer several questions. We examine the relationship between the real and approximated output sets. We also examine the relationship between the hyperplane and the approximated output sets, as this is crucial to the solving of the fault detection problem. One problem that we run into is that in the approximated systems, we do not have uniform convergence with respect to the initial functions, ϕ_i. However, by noting that our operators are compact, we will work around this problem to get uniformity in our results.

5.3.1 Review of Familiar Theorems and Definitions

There are several theorems and definitions which we will make use of in proving our main results. We begin with the definition of a *Compact Linear Operator.* As the spaces we are working in are not finite dimensional, by noting that our operators are compact linear operators, we can make infinite sets retain some of the properties of finite dimensional ones.

Definition 1 (Compact Linear Operator). *[41, pp. 405] Let X and Y be Banach spaces. An operator $T : X \to Y$ is called a* compact linear operator *(or, a* completely continuous linear operator*) if T is linear and if for every bounded subset M of X, the image $T(M)$ is relatively compact, that is, the closure $\overline{T(M)}$ is compact.*

Once we have defined a compact linear operator, we recall the theorem commonly referred to as the *Compactness Criterion.* As the name suggests, this theorem establishes criteria for determining if an operator is compact.

Theorem 4 (Compactness Criterion). *[41, pp. 407] Let X and Y be Banach spaces and $T : X \to Y$ a linear operator. Then T is compact if and only if it maps every bounded sequences $\{x_n\}$ onto a sequence $\{Tx_n\}$ in Y which has a convergent subsequence.*

Next, we recall two familiar definitions. The first is that of a *uniformly bounded family of continuous functions* and the second is *equicontinuous.* In a moment, we will recall the *Ascoli-Arzela Lemma* and these definitions play a key role in the lemma.

Definition 2. *Let $\mathcal{F}$ be a family of continuous functions defined on a set $E \subset \mathbf{R}$ with values in $\mathbf{R}^n$[1, pp. 22]. Then*

1. *$\mathcal{F}$ is called* uniformly bounded *if there is a nonnegative constant M such that $|f(t)| \leq M$ for all $t \in E$and for all $f \in \mathcal{F}$.*
2. *$\mathcal{F}$ is called* equicontinuous *on E if for every $\epsilon > 0$ if there is a $\delta = \delta(\epsilon) > 0$ (independent of t_1, t_2 and f) such that $|f(t_1) - f(t_2)| < \epsilon$ whenever $|t_1 - t_2| < \delta$ for all $t_1, t_2 \in E$ and for all $f \in \mathcal{F}$.*

Theorem 5 (Ascoli-Arzela Lemma). *[1, pp. 22] Let E be a closed and bounded subset of $\mathbf{R}$ and let $\{f_n\}$ be a sequence of functions in $C(E, \mathbf{R}^n)$. If $\{f_n\}$ is equicontinuous and uniformly bounded on E, then there is a subsequence $\{n_k\}$ of $\{n\}$ and a function $f \in C(E, \mathbf{R}^n)$ such that $\{f_{n_k}\}$ converges uniformly to f on E.*

We note that unlike many statements of the Ascoli-Arzela Lemma, we state this version for $\{f_n\}$ a sequence of functions in $C(E, \mathbf{R}^n)$. The usual proof of the lemma easily extends once it is noted that the Bolzano-Weirstrauss theorem generalizes to $\mathbf{R}^n$ by iterating in the following way. If $\{x_n\}$ is a bounded sequence in $\mathbf{R}^n$, then $\{x_{n_1}\}$ (the first element from each x_i) is a bounded sequence of real numbers and has a convergent subsequence. Call $\{x_{1n}\}$ the subsequence of $\{x_n\}$ defined by the convergent subsequence from $\{x_{n_1}\}$. Then $\{x_{1n}\}$ is a sequence of $\mathbf{R}^n$ in which the first "row" is a convergent sequence. Then $\{x_{1n_2}\}$ is a bounded sequence of real numbers and has a convergent subsequence. Thus, we can define a sequence, $\{x_{2n}\}$, which is subsequence of $\{x_n\}$ with the first two "rows" as convergent sequences. Proceeding similarly through all the n "rows", we can create a subsequence of $\{x_n\}$ which is convergent. Creating the subsequences in this way, the usual proof of the Ascoli-Arzela Lemma holds.

5.3.2 Some Useful Lemmas

Using these common theorems and definitions, we will build a store of lemmas for later use. By proving these lemmas now, we hope to make the proofs of our main theorems more transparent. Some of the lemmas are short and, perhaps, trivial; however, they are included for completeness. Other lemmas will play a key role in our main results.

The first lemma we prove has to do with the compactness of the operator $T(\mu) = \int_a^t k(s,t)\mu(s)ds$.

Lemma 4. *Let $T : L^2([a,b], \mathbf{R}^n) \to L^2([a,b], \mathbf{R}^n)$ be defined by $T(\mu) = \int_a^t k(s,t)\mu(s)ds$, $t \in [a,b]$, with $|k(s,t)| < K$ for some $K \in \mathbf{R}$ and all $s,t \in [a,b]$. Then T is a compact linear operator.*

Proof. Let $M = \{\mu |\ \mu \in L^2([a,b], \mathbf{R}^n), \|\mu\| \leq Q,$ for some fixed $Q \in \mathbf{R}\}$. By the Compactness Criterion, if T maps every bounded sequence $\{\mu_n\}$ in M onto a sequence $\{T(\mu_n)\}$ in $L^2[a,b]$ with a convergent subsequence, then T is a compact linear operator.

Let $\{\mu_n\}$ be a bounded sequence in M. Then $\{T(\mu_n)\}$ is a sequence in $L^2([a,b], \mathbf{R}^n)$. If we can show that $\{T(\mu_n)(t)\}$ is equicontinuous and uniformly bounded on $[a,b]$ for all n, then by the Ascoli-Arzela Lemma there exists a subsequence $\{T(\mu_{n_l})\}$ and a functional f in $C([a,b], \mathbf{R}^n)$ such that $\{T(\mu_{n_l})\}$ converges uniformly to f on $[a,b]$. Of course, since such an f is in $C([a,b], \mathbf{R}^n)$, then f is in $L^2([a,b], \mathbf{R}^n)$. Then $\{T(\mu_n)\}$ has a uniformly convergent subsequence in the L^2 sense and, thus, T is compact.

First, we show that $\{T(\mu_n)\}$ is uniformly bounded for each n.

$$\begin{aligned}
|(T\mu_n)(t)| &= \left|\int_a^t k(s,t)\mu_n(s)ds\right|, \ t \in [a,b] \\
&\leq \int_a^t |k(s,t)\mu_n(s)| \ ds, \ t \in [a,b] \\
\text{by Holders Inequality} \quad &\leq \left(\int_a^t |k(s,t)|^2 ds\right)^{\frac{1}{2}} \left(\int_a^t |\mu_n(s)|^2 ds\right)^{\frac{1}{2}}, \ t \in [a,b] \\
&\leq \left(\int_a^b |k(s,t)|^2 ds\right)^{\frac{1}{2}} \left(\int_a^b |\mu_n(s)|^2 ds\right)^{\frac{1}{2}} \\
&\leq \left(\int_a^b K^2 ds\right)^{\frac{1}{2}} Q \\
&\leq QK(b-a)^{\frac{1}{2}}.
\end{aligned}$$

Thus, $T(\mu_n)$ is uniformly bounded for all μ_n.

Next, we show that $T(\mu_n)$ is equicontinuous. Let $\epsilon > 0$ be given. Suppose $|t_1 - t_2| < \delta = \frac{\epsilon^2}{QK^2}$, $t_1, t_2 \in [a,b]$ with $t_2 > t_1$. Then

$$\begin{aligned}
|T(\mu_n)(t_1) - T(\mu_n)(t_2)| &= \left|\int_a^{t_1} k(s,t)\mu_n(s)ds - \int_a^{t_2} k(s,t)\mu_n(s)ds\right| \\
&= \left|\int_{t_1}^{t_2} k(s,t)\mu_n(s)ds\right| \\
&\leq \int_{t_1}^{t_2} |k(s,t)\mu_n(s)|ds \\
&\leq \left(\int_{t_1}^{t_2} |k(s,t)|^2 ds\right)^{\frac{1}{2}} \left(\int_{t_1}^{t_2} |\mu_n(s)|^2 ds\right)^{\frac{1}{2}} \\
&\leq \left(\int_{t_1}^{t_2} |k(s,t)|^2 ds\right)^{\frac{1}{2}} \left(\int_a^b |\mu_n(s)|^2 ds\right)^{\frac{1}{2}} \\
&\leq \left(\int_{t_1}^{t_2} |k(s,t)|^2 ds\right)^{\frac{1}{2}} Q \\
&\leq \left(\int_{t_1}^{t_2} K^2 ds\right)^{\frac{1}{2}} Q \\
&\leq \left(K^2(t_2 - t_1)\right)^{\frac{1}{2}} Q \\
&\leq \left(K^2\delta)\right)^{\frac{1}{2}} Q \\
&\leq \epsilon
\end{aligned}$$

Thus, $T(\mu_n)$ is equicontinuous and T is compact. $\diamondsuit$

Our next lemma is a simple consequence of the definition of a compact set.

Lemma 5. *If B is a compact set, then for any given $\epsilon > 0$, there exists $K_1(\epsilon)$ and $\{b_1, b_2, \ldots, b_{K_1(\epsilon)}\} \subset B$ such that for all $b \in B$, $\|b - b_i\| < \epsilon$ for some i in $i = 1, 2, \ldots, K_1(\epsilon)$.*

Proof. Suppose B is a compact set. Let $\epsilon > 0$ be given. Let B_ϵ be the open covering of B consisting of open balls, $B_\epsilon(b)$ of size ϵ around each point $b \in B$. Then B_ϵ has a finite uncovering. This implies there exists $b_1, b_2, \ldots, b_{K_1(\epsilon)}$ such that for all $b \in B$, $b \in \bigcup_{i=1}^{K_1(\epsilon)} B_\epsilon(b_i)$. Then for all $b \in B$, there exists some $j \in 1, \ldots, K_1(\epsilon)$ such that $|b - b_j| < \epsilon$. $\diamondsuit$

Our next lemma shows that the sum of two compact linear operators is also a compact linear operator. Although this might seem obvious, notice that the two operators have different domains. This fact makes this lemma somewhat different than what is typically seen in Analysis texts.

Lemma 6. *Let $T_1 : X_1 \to Y$ and $T_2 : X_2 \to Y$ be compact linear operators. Then $T : X_1 \times X_2 \to Y$ defined by $T(x_1, x_2) = T_1(x_1) + T(x_2)$ is also a compact linear operator.*

Proof. Let $\widetilde{T_1} : X_1 \times X_2 \to Y$ be defined by $\widetilde{T_1}(x_1, x_2) = T_1(x_1)$. Since T_1 is compact linear operators, then for every bounded subset B_1 of X_1, $\overline{T_1(B_1)}$ is compact. Then for every bounded subset B of $X_1 \times X_2$, $\overline{\widetilde{T_1}(B)}$ is compact and $\widetilde{T_1}$ is a compact linear operator.

Let $\widetilde{T_2} : X_1 \times X_2 \to Y$ be defined by $\widetilde{T_2}(x_1, x_2) = T_1(x_1)$. Since T_2 is compact linear operators, then for every bounded subset B_2 of X_2, $\overline{T_2(B_2)}$ is compact. Then for every bounded subset B of $X_1 \times X_2$, $\overline{\widetilde{T_2}(B)}$ is compact and $\widetilde{T_2}$ is a compact linear operator.

Then $T = \widetilde{T_1} + \widetilde{T_2}$ is the sum of two compact linear operators and T is a compact linear operator. $\diamondsuit$

Lemmas 7 and 8 will play an important role in our main results. By using the fact that the closure of the image of a bounded set is compact if the operator is compact, we can, in some sense, pull back the compactness to our domain. We will use this to show that there exists a finite number of (x_{0_i}, ϕ_i, μ_i) in our permissible initial data set so that given any solution x to the delayed system, for some i, $\|x(t, x_0, \phi, \mu) - x(t, x_{0_i}, \phi_i, \mu_i))\| < \delta$ for some given δ.

Lemma 7. *Let $M = \{x \mid \ \|x\| < c,\ x \in X,$ a normed space$\}$. If T is a compact operator on M then for any given $\epsilon > 0$, there exists $K(\epsilon)$ and $\{x_1, x_2, \ldots, x_{K(\epsilon)}\} \subset M$ such that for all $x \in M$, $\|T(x) - T(x_i)\| < \epsilon$ for* at least *one $i = 1, 2, \ldots, K(\epsilon)$.*

Proof. Suppose T is a compact operator on M. This implies that $\overline{T(M)}$ is a compact set. Let $B = \bigcup_{x\in M} B_{T(x)}$ be the open cover of $\overline{T(M)}$ consisting of open balls, $B_{T(x)}$, of radius ϵ about the elements of $T(M)$.

Note that this does cover $\overline{T(M)}$. It clearly covers $T(M)$. It covers any accumulation point of $T(M)$, since by definition, if z is an accumulation point of $T(M)$, then every neighborhood of z contains an element of $T(M)$ other than x. Thus, an $\frac{\epsilon}{2}$ ball about z contains some element $b \neq z$ in $T(M)$. But, this would imply that the ϵ ball about b contains z. Thus, B contains $\overline{T(M)}$.

Since B is an open cover of $\overline{T(M)}$, then B admits a finite subcover. Since the open balls are all centered around points in $T(M)$, then there must be a $K(\epsilon)$ and $x_1, x_2, \ldots, x_{K(\epsilon)}$ such that for any $x \in M$, $\|T(x) - T(x_i)\| < \epsilon$ for at least one i in $\{1, 2, \ldots, K(\epsilon)\}$. $\diamondsuit$

Lemma 8. *Let $M_1 = \{x_0 \mid |x_0| < 1,\ x \in \mathbf{R}^n\}$, $M_2 = \{\phi \mid \ \|\phi\| < 1,\ \phi \in L^2([a, b], \mathbf{R}^n)\}$, and $M_3 = \{\mu \mid \|\mu\| < 1,\ \mu \in L^2([a, b], \mathbf{R}^n)\}$. Let $M = \{(x_0, \phi, \mu) \in M_1 \times M_2 \times M_3 \mid x_0^T P x_0 + \|\phi\| + \|\mu\| < 1\}$ for P some positive, semi-definite, constant matrix. For any given $\delta > 0$, there exists $\{(x_0, \phi, \mu)_1, (x_0, \phi, \mu)_2, \ldots, (x_0, \phi, \mu)_{K1(\delta)}\} \subset M$ such that for all $(x_0, \phi, \mu) \in M$*

$$\|x(t, x_0, \phi, \mu) - x(t, x_{0_i}, \phi_i, \mu_i))\| < \delta$$

for some $i \in \{1, 2, \ldots, K1(\delta)\}$ $\}$.

Proof. Suppose

1. $L_1(x_0) = X(t)x_0$
2. $L_2(\phi) = \int_{-r}^{0} X(t - s - r)G\phi(s)ds$
3. $L_3(f) = \int_0^t X(t - s)Mf(s)ds$

4. $L_4(f) = \int_0^t X(t-s)Bf(s)ds$.

Then $x(t, x_0, \phi, \mu) = L_1(x_0) + L_2(\phi) + L_4(v) + L_3(\mu)$. We need to find (x_0^*, ϕ^*, μ^*) such that $x^*(t, x_0^*, \phi^*, \mu^*) = L_1(x_0^*) + L_2(\phi^*) + L_4(v) + L_3(\mu^*)$ for a fixed v and

$$\|x - x^*\| \leq \delta.$$

Let $T(x_0, \phi, \mu) = L_1(x_0) + L_2(\phi) + L_3(\mu)$.

For any x^*,

$$\begin{aligned}
\|x - x^*\| &= \|L_1(x_0) + L_2(\phi) + L_4(v) + L_3(\mu) - (L_1(x_0^*) + L_2(\phi^*) + L_4(v) + L_3(\mu^*))\| \\
&\leq \|L_1(x_0) + L_2(\phi) + L_3(\mu) - (L_1(x_0^*) + L_2(\phi^*) + L_3(\mu^*))\| \\
&\leq \|T(x_0, \phi, \mu) - T(x_0^*, \phi^*, \mu^*)\|.
\end{aligned}$$

But, L_1 is a compact linear operator since any closed, bounded subset of $\mathbf{R}^n$ is compact. L_2 and L_3 are compact linear operators by Lemma 4. Thus, by Lemma 6, T is a compact linear operator.

Since T is a compact linear operator on $M_1 \times M_2 \times M_3$ and M is a bounded subset of $M_1 \times M_2 \times M_3$, then by Lemma 7, for any given $\delta > 0$, there exists

$$\{(x_0, \phi, \mu)_1, (x_0, \phi, \mu)_2, \ldots, (x_0, \phi, \mu)_{K1(\delta)}\} \subset M$$

such that for all $(x_0, \phi, \mu) \in M$

$$\|T(x_0, \phi, \mu) - T(x_{0_i}, \phi_i, \mu_i)\| < \delta$$

for some $i = 1 \ldots K1(\delta)$. $\diamondsuit$

Later we will make use of a convergence theorem from Ito and Kappel [34]. However, the statement of this theorem is for controls in $L^1[a,b]$, so here we take a moment to formally note that $L^2[a,b]$ is a subset of $L^1[a,b]$ since $[a,b]$ is a finite interval.

Lemma 9. *If $-\infty < a < b < \infty$, then $L^2[a,b] \subseteq L^1[a,b]$.*

Proof. Suppose $f \in L^2[a,b]$. Obviously, the constant function $\mathbf{1} \in L^2[a,b]$. Then by Holder's inequality,

$$\begin{aligned}
\int_a^b |f(t)\mathbf{1}| \;\; dt &\leq \|f\|_{L^2}\|\mathbf{1}\|_{L^2} \\
\int_a^b |f(t)| \;\; dt &\leq \|f\|_{L^2}\|\mathbf{1}\|_{L^2} \\
\|f\|_{L^1} &\leq \|f\|_{L^2}\|(b-a)^{\frac{1}{2}}.
\end{aligned}$$

Thus, $f \in L^2[a,b] \Rightarrow f \in L^1[a,b]$. ♢

Our next lemma shows a relationship between strong convergence in operators and the inner product. It will be useful in our results concerning approximated output sets and the hyperplane.

Lemma 10. *Suppose X and Y are Hilbert spaces. Suppose $T_n : X \to Y$ and $T : X \to Y$. If $T_n \to T$ strongly and $\|y\| < K$, then for any given ϵ there exists $N(x,\epsilon)$ such that*

$$< y, T_n x > \leq \;\; < y, Tx > +\epsilon$$

for $N \geq N(x,\epsilon)$.

Proof. Let ϵ be given and choose $\delta = \frac{\epsilon}{K}$.

$$\begin{aligned}
< y, T_n x > &= \;\; < y, (T_n - T + T)x > \\
&= \;\; < y, (T_n - T)x > + < y, Tx > \\
&\leq \;\; \|y\|\|(T_n - T)x\| + < y, Tx > \\
&\leq \;\; K\delta + < y, Tx > \quad \text{for some} \quad N \geq N(x,\epsilon) \\
&\leq \;\; K\frac{\epsilon}{K} + < y, Tx > \quad \text{for some} \quad N \geq N(x,\epsilon) \\
&\leq \;\; < y, Tx > +\epsilon \quad \text{for some} \quad N \geq N(x,\epsilon).
\end{aligned}$$

♢

Lemma 11 establishes a relationship between compact operators and strong convergence. As we will see, while some of the operators we are using only converge strongly, we will be able to get a type of uniform convergence nonetheless.

Lemma 11. *Suppose X and Y are Hilbert spaces. Suppose $S : X \to Y$ is a compact operator, $P^N : Y \to Y$ is bounded for each N with $P^N \to I$ strongly, and $M \subset X$ is bounded. Then $P^N S \to S$ uniformly.*

Proof. Suppose not. If not, then this implies that for any given ϵ, there exists $\{z_m\}$ such that

$$\|P^N S z_m - S z_m\| > \epsilon$$

for each m. But, since S is compact, by the Compactness Criterion (Theorem 4), S takes a bounded sequence, $\{z_m\}$, to a sequence, $\{Sz_m\}$, which has a convergent subsequence. Clearly, $\{z_m\} \subset M$ is bounded. Thus, there is a subsequence $\{z_{m_k}\}$ such that $\{Sz_{m_k}\}$ converges. Suppose $Sz_{m_k} \to \alpha$. This implies that

$$Sz_{m_k} = \alpha + \delta_k \text{ with } \delta_k \to 0.$$

Then

$$\begin{aligned} \|P^N S z_{m_k} - S z_{m_k}\| &= \|P^N(\alpha + \delta_k) - (\alpha + \delta_k)\| \\ &= \|(P^N - I)\alpha + (P^N - I)\delta_k\| > \epsilon \end{aligned}$$

for all k. But, $\delta_k \to 0$ and $(P^N - I)\alpha \to 0$. Thus, we have a contradiction. $\diamondsuit$

Our last lemma and its corollary note the relationship between bounded operators and sequences of operators which converge strongly.

Lemma 12. *Suppose X and Y are Hilbert spaces and $P^N : Y \to Y$, $T_n : X \to Y$, and $T : X \to Y$. If $\|P^N\| < K$ for all N and $T^N \to T$ strongly, then $P^N T^N \to P^N T$ strongly.*

Proof. Let ϵ be given and choose $\delta = \frac{\epsilon}{K}$. For all z,

$$\begin{aligned}
\|P^N T^N x - P^N T x\| &= \|P^N (T^N x - Tx)\| \\
&\leq \|P^N\| \|(T^N x - Tx)\| \\
&\leq K \|(T^N x - Tx)\| \\
&\leq K\delta \ \text{ for sufficiently large } N \\
&\leq K\frac{\epsilon}{K} \\
&\leq \epsilon.
\end{aligned}$$

$\diamondsuit$

Corollary 1. *Suppose X and Y are Hilbert spaces and that $Q : X \to Y$, $T_n : X \to Y$, and $T : X \to Y$. If $\|Q\| < K$ and $T^N \to T$ strongly then, $QT^N \to QT$ strongly.*

Proof. Let $P^N = Q$ for all N. Then by Lemma 12, $QT^N \to QT$ strongly. $\diamondsuit$

5.3.3 How Close Are the Real and Approximated Output Sets?

The first question we ask is 'How close are the real and approximated model output sets?' One way to measure this is to show that given a $y \in \mathcal{A}_i$, there is a $y^N \in \mathcal{A}_i^N$ close to it. Another question we ask is, "Is this closeness uniform in some way?" As is shown in Theorem 6, the answer to the question is 'yes'.

Theorem 6. *Suppose we have the system (5.20). If $x_0^t P x_0 + \|\phi\| + \|\mu\| \leq 1$, then there exists $\bar{N}(\epsilon)$ such that for any $\epsilon > 0$, and for any $y \in \mathcal{A}_i(v)$, there is a $y^N \in \mathcal{A}_i^N(v)$ such that $\|y - y^N\| < \epsilon$ whenever $N > \bar{N}(\epsilon)$.*

Proof. Let $M_1 = \{x \mid \|x\| \leq 1,\ x \in \mathbf{R}^n\}$, $M_2 = \{\phi \mid \ \|\phi\| \leq 1,\ \phi \in L^2([a,b], \mathbf{R}^n)\}$, and $M_3 = \{\mu_2 \mid \|\mu_2\| \leq 1,\ \mu_2 \in L^2([a,b], \mathbf{R}^n)\}$. Let $M = \{(x_0, \phi, \mu_2) \in M_1 \times M_2 \times M_3 \mid x_0^T P x_0 + \|\phi\| + \|\mu_2\| \leq 1\}$.

We wish to show that there exists $\bar{N}(\epsilon)$ such that for any $\epsilon > 0$, if $N > \bar{N}(\epsilon)$, for any $y \in \mathcal{A}_i(v)$, there is some $y^N \in \mathcal{A}_i^N(v)$ such that $\|y - y^N\| < \epsilon$.

Now, as

$$\begin{aligned}\|y(t,x_0,\phi,\mu)-y(t,x_0,\phi,\mu)^N\| &= \|C(x-x^N)+N(\mu-\mu^N)\|\\ &\leq |C|\|(x-x^N)\|\tilde{N}(\mu_1-\mu_1^N)\|\\ &\leq |C|\|(x-x^N)\|+|\tilde{N}|\|(\mu_1-\mu_1^N)\|.\end{aligned}$$

Since we can consider any value of μ_1^N, as N is not dependent on the value μ_1^N we choose $\mu_1^N=\mu_1$ and $\|(\mu_1-\mu_1^N)\|=0$.

Now, suppose that $\epsilon>0$ is given and $\delta=\frac{\epsilon}{2|C|}$. Assume for the moment that there exists $\{(x_0,\phi,\mu_2)_1,(x_0,\phi,\mu_2)_2,\ldots,(x_0,\phi,\mu_2)_{K1(\epsilon)}\}\subset M$ such that for all $(x_0,\phi,\mu_2)\in M$

$$\|x(t,x_0,\phi,\mu_2)-x(t,x_{0_i},\phi_i,\mu_{2_i})\|<\delta$$

for some $i\in\{1,2,\ldots,K1(\epsilon)\}$.

Now, we know that [34, pp. 233] for all x, $\|x-x^N\|\to 0$ as $N\to\infty$. Then for each $(x_0,\phi,\mu_2)_i$, there exists an N_i such that

$$\|x(t,x_{0_i},\phi_i,\mu_{2_i})-x^N(t,x_{0_i},\phi_i,\mu_{2_i})\|<\delta$$

whenever $N>N_i$. Thus, for $\overline{N}(\epsilon)=\max_i\{N_i\}$ with $i\in\{1,2,\ldots,K1(\epsilon)\}$

$$\|x(t,x_{0_i},\phi_i,\mu_{2_i})-x^N(t,x_{0_i},\phi_i,\mu_{2_i})\|<\delta$$

for all i whenever $N>\overline{N}(\epsilon)$.

Then for all x there is an $x_i=x(t,x_{0_i},\phi_i,\mu_{2_i})$ and $x_i^N=x^N(t,x_{0_i},\phi_i,\mu_{2_i})$ such that

$$\begin{aligned}|C|\|x-x_i^N\| &= |C|\|(x-x_i+x_i-x_i^N)\|\\ &\leq |C|(\|x-x_i\|+\|x_i-x_i^N\|)\\ &\leq |C|(\delta+\delta)\\ &\leq |C|(\frac{\epsilon}{2|C|}+\frac{\epsilon}{2|C|})\\ &\leq \epsilon\end{aligned}$$

whenever $N>\overline{N}(\epsilon)$. Now, as $N(\epsilon)$ was independent of μ_1, for any y, we can form $\mu_i=\begin{bmatrix}\mu_1\\ \mu_{2_i}\end{bmatrix}$ and

$$\|y(t,x_0,\phi,\mu)-y(t,x_{0_i},\phi_i,\mu_i)_i^N\| \quad\leq\quad \epsilon$$

whenever $N > \overline{N}(\epsilon)$ for some i in $1 \ldots K1(\epsilon)$.

Thus, it remains to show only that there exists $\{(x_0, \phi, \mu_2)_1, (x_0, \phi, \mu_2)_2, \ldots, (x_0, \phi, \mu_2)_{K1(\epsilon)}\} \subset M$ such that for all $(x_0, \phi, \mu_2) \in M$

$$\|x(t, x_0, \phi, \mu_2) - x(t, x_{0_i}, \phi_i, \mu_{2_i})\| < \delta$$

for some $i \in \{1, 2, \ldots, K1(\epsilon)\}$ such that $x_{0_i}^t P x_{0_i} + \|\phi_i\| + \|\mu\| \leq 1$.

We know that $y = Cx + N\mu = Cx + \tilde{N}\mu_1$ with $\|\mu_1\| = \alpha$ for some $0 \leq \alpha \leq 1$. Then as $x_0^t P x_0 + \|\phi\| + \|\mu\| \leq 1$, this implies that $x_0^t P x_0 + \|\phi\| + \|\mu_2\| \leq \sqrt{1 - \alpha^2}$. Let $c = \sqrt{1 - \alpha^2}$.

Consider $\hat{x} = \frac{1}{c}x = x(t, \frac{1}{c}x_0, \frac{1}{c}\phi, \frac{1}{c}\mu_2)$. Clearly, if $x_0^t P x_0 + \|\phi\| + \|\mu_2\| \leq c$ then $\frac{1}{c}x_0^t P \frac{1}{c}x_0 + \|\frac{1}{c}\phi\| + \|\frac{1}{c}\mu_2\| \leq 1$. Then by Lemma 8, for any given $\delta > 0$, we can find

$$\{(x_0, \phi, \mu_2)_1, (x_0, \phi, \mu_2)_2, \ldots, (x_0, \phi, \mu_2)_{K1(\epsilon)}\} \subset M$$

such that for all $(x_0, \phi, \mu_2) \in M$,

$$\|\hat{x}(t, x_0, \phi, \mu_2) - x(t, x_{0_i}, \phi_i, \mu_{2_i})\| < \delta$$

for some $i \in \{1, 2, \ldots, K1(\epsilon)\}$. But, this implies that

$$\|x(t, x_0, \phi, \mu_2) - x(t, cx_{0_i}, c\phi_i, c\mu_{2_i}))\| < c\delta \leq \delta$$

as $c \leq 1$. And, as $x_{0_i}^t P x_{0_i} + \|\phi_i\| + \|\mu_{2_i}\| \leq 1$, then $cx_{0_i}^t P c x_{0_i} + \|c\phi_i\| + \|c\mu_{2_i}\| \leq c$, so $x_{0_i}^t P x_{0_i} + \|\phi_i\| + \|\mu_i\| \leq 1$. $\diamondsuit$

5.3.4 A Hyperplane Which Separates the Real Sets Separates the Approximated Sets

After the detection signal is found, the next step is to determine the separating hyperplane, as discussed in Section 1.3.2. In this section, we show that a hyperplane which separates the true output sets by a non-zero amount will also separate the approximated output sets.

Theorem 7. *Suppose we have the system (5.20). If $x_0^t P x_0 + \|\phi\| + \|\mu\| \leq 1$ and $\mathcal{A}_i(v)$ is the output set of model i, $\mathcal{A}_i^N(v)$ is the approximated output set of model i, with a as any supporting hyperplane such that for all $y \in \mathcal{A}_i(v)$,*

$$< y, a > \geq \quad \epsilon.$$

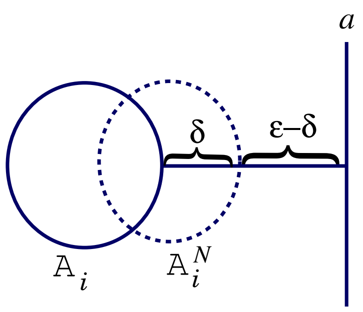

Figure 5.2: If a hyperplane separates the real output sets, then it also separates the approximated output sets. More specifically, we show that given $< y, a > \geq \ \ \epsilon$ where a is a hyperplane, then for any $\delta > 0$, there exists $\overline{N}$ such that if $n > \overline{N}$ then for all $y^N \in \mathcal{A}_i^N(v)$, $< y^N, a > \geq \ \ \epsilon - \delta$.

Then for any $\delta > 0$, there exists $\overline{N}$ such that if $n > \overline{N}$ then for all $y^N \in \mathcal{A}_i^N(v)$,

$$< y^N, a > \geq \ \ \epsilon - \delta.$$

Proof. Take some $y^N(t, x_0^N, \mu_1^N, v, \mu_2^N) \in \mathcal{A}_i^N(v)$. Then

$$\begin{aligned} < y^N, a > &= < y, a > -(< y, a > - < y^N, a >) \ \ for \ \ any \ \ y \in \mathcal{A}_i(v) \\ &\geq \epsilon - (< y, a > - < y^N, a >) \\ &\geq \epsilon - (< C(x - x^N) + N(\mu_1 - \mu_1^N), a >). \end{aligned}$$

Thus, we need to find $\overline{N}$ such that for all $y^N \in \mathcal{A}_i^N(v)$, if $N > \overline{N}$, then

$$< C(x - x^N) + N(\mu_1 - \mu_1^N), a > \ < \ \delta$$

for some $y \in \mathcal{A}_i(v)$.

Let us consider

$$\begin{aligned} < C(x - x^N) + N(\mu_1 - \mu_1^N), a > &= < C(x - x^N), a > + < N(\mu_1 - \mu_1^N), a > \\ &\leq < C(x - x^N), a > + |N| \|\mu_1 - \mu_1^N\| \|a\|. \end{aligned}$$

Thus far, the y (and consequently, x and μ) have been arbitrary. That is, what we have said is true of any y in $\mathcal{A}_i(v)$. Since we can consider any value of μ_1, as N is not dependent on the value μ_1^N, we choose $\mu_1 = \mu_1^N$ and $\|\mu_1 - \mu_1^N\| = 0$. Suppose, for the moment, that we can find a particular y or finite subset of y's such that $< C(x - x^N), a >< \delta$. Then we would have

$$\begin{aligned} < C(x - x^N) + N(\mu_1 - \mu_1^N), a > &= < C(x - x^N), a > + |N| \|\mu_1 - \mu_1^N\| \|a\| \\ &< \delta \end{aligned}$$

and we are done.

Thus, it remains to show that we can find a y such that $< C(x - x^N), a >< \delta$.

For convenience, we change variables, using the variables as described in [34] and Section 5.1. Note that $\mathcal{M}^N = \mathcal{M}$, $\mathcal{C}^N = \mathcal{C}$, and $\mathcal{B}^N = \mathcal{B}$. Suppose

1. $L_1(z_0) = S(t)z_0$
2. $L_1^N(z_0) = S^N(t)P^N z_0$
3. $L_2(\phi) = \int_0^t S(t-s)\mathcal{M}f(s)ds$
4. $L_2^N(\phi) = \int_0^t S^N(t-s)\mathcal{M}f(s)ds$
5. $L_3(f) = \int_0^t S(t-s)\mathcal{B}f(s)ds$
6. $L_3^N(f) = \int_0^t S^N(t-s)\mathcal{B}f(s)ds$

where S^N is the semigroup on Z^N generated by $\mathcal{A}^\mathcal{N}$, i.e., $S^N(t) = e^{\mathcal{A}^N t}$, in the approximated system. Let $a = \begin{bmatrix} a \\ 0 \end{bmatrix}$ so that the dimensions are appropriate for our problem. Then

$$\begin{aligned} < C(x - x^N), a > &= < C(z - z^N), a > \\ &= < C((L_1(z_0) + L_2(\mu) + L_3(v)) - (L_1^N(z_0) + L_2^N(\mu) + L_3^N(v))), a > \\ &= < C(L_1(z_0) - L_1^N(z_0) + (L_2(\mu) + L_3(v) - (L_2^N(\mu) + L_3^N(v))), a > \\ &= < C(L_1(z_0) - L_1^N(z_0)), a > \\ &\quad + < C(L_2(\mu) + L_3(v) - (L_2^N(\mu) + L_3^N(v))), a > . \end{aligned}$$

We note that by:

By theorem 3.4 in [34, pp. 232], we have that $S^{N*}P^N - S^* \to 0$ strongly. Then by Lemma 12,

$$P^N S^{N*} P^N - P^N S^* \to 0 \quad \text{strongly.} \tag{5.36}$$

Also, by Lemma 11, we have

$$P^N S^* - S^* \to 0 \quad \text{uniformly} \tag{5.37}$$

and by Lemma 1

$$S^* P^N - S^* \to 0 \quad \text{strongly.} \tag{5.38}$$

Now, let us consider $< C(L_1(z_0) - L_1^N(z_0)), a >$. We have

$$\begin{aligned}
< (L_1(z_0) - L_1^N(z_0)), a > &= < S(t)z_0 - S^N(t)P^N z_0, C^* a > \\
&= < (S(t) - S^N(t)P^N)z_0, C^* a > \\
&\leq < (S(t) - S^N(t)P^N)z_0, P^N C^* a > + \delta_1 \\
&\quad\ \text{for } N > N(\delta_1, a) \ \text{ by Lemma 10 since } P^N \to I \\
&\leq < z_0, (S^*(t) - P^{N*}S^{N*}(t))P^N C^* a > + \delta_1 \\
&\quad\ \text{for } N > N(\delta_1, a) \\
&\leq < z_0, (S^*(t)P^N - P^N S^{N*}(t)P^N)C^* a > + \delta_1 \\
&\quad\ \text{for } N > N(\delta_1, a) \\
&\leq < z_0, (S^*(t) - P^N S^{N*}(t)P^N)C^* a > + \delta_1 + \delta_2 \\
&\quad\ \ \text{for } N > \max\{N(\delta_1, a), N(\delta_2)\} \\
&\quad\ \text{by Lemma 10 and (5.38)} \\
&\leq < z_0, (P^N S^*(t) - P^N S^{N*}(t)P^N)C^* a > + \delta_1 + \delta_2 + \delta_3 \\
&\quad\ \ \text{for } N > \max\{N(\delta_1, a), N(\delta_2), N(\delta_3, a)\} \\
&\quad\ \text{by Lemma 10 and (5.37)} \\
&\leq \|z_0\| \|(P^N S^*(t) - P^N S^{N*}(t)P^N)C^* a\| + \delta_1 + \delta_2 + \delta_3 \\
&\leq \delta_1 + \delta_2 + \delta_3 + \delta_4 \\
&\quad\ \text{for } N > \max\{N(\delta_1, a), N(\delta_2), N(\delta_3, a), N(\delta_4, a)\} \\
&\quad\ \text{by (5.36).}
\end{aligned}$$

Thus, by choosing $\delta_1, \delta_2, \delta_3$ and δ_4 such that their sum is less than $\frac{\delta}{2}$, we have

$$< C(L_1(z_0) - L_1^N(z_0)), a > \ \leq \ \frac{\delta}{2}$$

for all z_0, whenever $N > \max\{N(\delta_1, a), N(\delta_2), N(\delta_3, a), N(\delta_4, a)\}$.

Now, we consider $< C(L_2(\mu) + L_3(v) - (L_2^N(\mu) + L_3^N(v)), a >$. We note that $L_2(\mu) + L_3(v) - (L_2^N(\mu) + L_3^N(v)))$ represents the difference in solutions to the actual and approximated problem with 0 initial condition and 0 initial function. We know by Corollary 3.5 in [34, pp. 233], all solutions converge in the controls uniformly, for controls in bounded subsets of $L_1[0, T]$. Our sets are bounded in L^2; but, we know by Lemma 9 that bounded in L^2 implies bounded in L^1 (on a finite interval). So, for all $\delta > 0$ there is an $\hat{N}$ such that if $N > \hat{N}$, then

$$\begin{aligned} < C(L_2(\mu) + L_3(v) - L_2^N(\mu) + L_3^N(v)), a > \ &\leq \ |C| \|(L_2(\mu) + L_3(v) \\ &\quad -(L_2^N(\mu) + L_3^N(v)))\| \|a\| \\ &\leq \ \frac{\delta}{2|C|\|a\|}. \end{aligned}$$

Thus, we have for $N > \max\{\hat{N}, N(\delta_1, a), N(\delta_2, a), N(\delta_3, a), N(\delta_4, a))\}$,

$$\begin{aligned} < C(x - x^N), a > \ &= \ < C(z - z^N), a > \\ &= \ < C(L_1(z_0) - L_1^N(z_0)), a > \\ &\quad + < C(L_2(\mu) + L_3(v) - (L_2^N(\mu) + L_3^N(v))), a > \\ &< \ \frac{\delta}{2} + \frac{\delta}{2} \\ &< \ \delta. \end{aligned}$$

$\diamondsuit$

As a corollary to the previous theorem, we have that a strictly proper, approximated detection signal for the real models is also strictly proper for the approximated models.

Corollary 2. *Suppose we have the system (5.20). Suppose $v^N \to v$, $\mathcal{A}_i(v^N)$ is the output set of model i, and $\mathcal{A}_i^N(v^N)$ is the approximated output set of model i, with a as any supporting hyperplane such that for all $y \in \mathcal{A}_i(v^N)$,*

$$< y, a > \geq \ \ \epsilon \quad \textit{whenever } N > N(\epsilon).$$

Then for any $\delta > 0$, there exists $\overline{N}$ such that if $n > \overline{N}$ then for all $y^N \in \mathcal{A}_i^N(v^N)$,

$$< y^N, a > \geq \quad \epsilon - \delta.$$

Proof. Let

$$\begin{aligned} y^N &= y^N(t, x_0^N, \mu_1^N, v^N, \mu_2^N) \in \mathcal{A}_i^N(v^N), \\ y &= y(t, x_0, \mu_1, v^N, \mu_2) \in \mathcal{A}_i(v^N), \\ y^N(v) &= y^N(t, x_0^N, \mu_1^N, v, \mu_2^N) \in \mathcal{A}_i^N(v), \end{aligned}$$

and

$$y(v) \quad = \quad y(t, x_0, \mu_1, v, \mu_2) \in \mathcal{A}_i(v).$$

Then

$$\begin{aligned} < y^N, a > &= \; < y, a > -(< y, a > - < y^N, a >) \text{ for any } \; y \in \mathcal{A}_i(v^N) \\ &\geq \; \epsilon - (< y, a > - < y^N, a >) \text{ whenever } N > N(\epsilon). \end{aligned}$$

Thus, we need to find $\overline{N}$ such that for all $y^N \in \mathcal{A}_i^N(v^N)$, if $N > \overline{N}$, then

$$< y - y^N, a > \; < \; \delta$$

for some $y \in \mathcal{A}_i(v^N)$. For convenience, we change variables, using the variables as described in [34] and Section 5.1. Note that $\mathcal{M}^N = \mathcal{M}$, $\mathcal{C}^N = \mathcal{C}$, and $\mathcal{B}^N = \mathcal{B}$. Suppose

1. $L_1(z_0) = S(t)z_0$
2. $L_1^N(z_0) = S^N(t)P^N z_0$
3. $L_2(\phi) = \int_0^t S(t-s)\mathcal{M}f(s)ds$
4. $L_2^N(\phi) = \int_0^t S^N(t-s)\mathcal{M}f(s)ds$
5. $L_3(f) = \int_0^t S(t-s)\mathcal{B}f(s)ds$
6. $L_3^N(f) = \int_0^t S^N(t-s)\mathcal{B}f(s)ds$

where S^N is the semigroup on Z^N generated by $\mathcal{A}^{\mathcal{N}}$, i.e., $S^N(t) = e^{\mathcal{A}^N t}$, in the approximated system. Let $a = \begin{bmatrix} a \\ 0 \end{bmatrix}$ so that the dimensions are appropriate for our problem. Then

$$\begin{aligned}
< y - y^N, a > &= < C(z - z^N), a > + < N(\mu_1 - \mu_1^N), a > \\
&= < C[(L_1(z_0) + L_2(\mu) + L_3(v^N)) \\
&\quad -(L_1^N(z_0) + L_2^N(\mu) + L_3^N(v^N))], a > + < N(\mu_1 - \mu_1^N), a > \\
&= < C[(L_1(z_0) + L_2(\mu) + L_3(v^N)) - (L_1^N(z_0) + L_2^N(\mu) + L_3^N(v^N)) \\
&\quad +L_3^N(v) - L_3^N(v) + L_3(v) - L_3(v)], a > + < N(\mu_1 - \mu_1^N), a > \\
&= < C[(L_1(z_0) + L_2(\mu) + L_3(v)) - (L_1^N(z_0) + L_2^N(\mu) + L_3^N(v)) \\
&\quad +L_3^N(v^N) - L_3^N(v) + L_3(v^N) - L_3(v)], a > + < N(\mu_1 - \mu_1^N), a > \\
&= < (y(v) - y^N(v)), a > + < C(L_3^N(v^N - v) + L_3(v^N - v)), a > \\
&= < (y(v) - y^N(v)), a > + \| C(L_3^N(v^N - v) + L_3(v^N - v))\| \|a\|.
\end{aligned}$$

But, we have by assumption that $v^N \to v$ and since we are working on a closed and bounded interval, there exists an $N_1(\frac{\delta}{2})$ such that for $N > N_1(\frac{\delta}{2})$, $\|C(L_3^N(v^N - v) + L_3(v^N - v))\|\|a\| < \frac{\delta}{2}$. And, from the proof of our previous theorem, we know we can make $< (y(v) - y^N(v)), a > < \frac{\delta}{2}$ for some $N_2(\frac{\delta}{2})$. Thus, we have

$$\begin{aligned}
< (y(v) - y^N(v)), a > + < (L_3^N(v^N - v) + L_3(v^N - v)), a > &< \; < (y(v) - y^N(v)), a > + \frac{\delta}{2} \\
&< \; \frac{\delta}{2} + \frac{\delta}{2} \\
&< \; \delta \text{ for} \\
&\qquad \overline{N} > \max\{N_1(\frac{\delta}{2}), N_2(\frac{\delta}{2})\}.
\end{aligned}$$

Then we have that

$$< y^N, a > \geq \quad \epsilon - \delta \quad \text{whenever } N > \max\{N_1(\frac{\delta}{2}), N_2(\frac{\delta}{2}), N(\epsilon)\}.$$

$\diamondsuit$

5.3.5 A Hyperplane Which Separates the Approximated Sets Separates the Real Sets

Another question of interest to us is whether a hyperplane which separates the approximated output sets will separate the real output sets. In the next theorem, we see that this is the case.

Theorem 8. *If $x_0^t P x_0 + \|\phi\| + \|\mu\| \leq 1$ and if $\mathcal{A}_i(v)$ is the output set of model i, $\mathcal{A}_i^N(v)$ is the approximated output set of model i, with a as any supporting hyperplane such that for all $y^N \in \mathcal{A}_i^N(v)$,*

$$< y^N, a > \geq \quad \epsilon \quad \textit{whenever } N > N(\epsilon).$$

Then for any $\delta > 0$, there exists $\overline{N}$ such that if $n > \overline{N}$ then for all $y \in \mathcal{A}_i(v)$,

$$< y, a > \geq \quad \epsilon - \delta.$$

Proof. Take some $y \in \mathcal{A}_i(v)$. Then

$$\begin{aligned} < y, a > &= \; < y^N, a > -(< y^N, a > - < y, a >) \;\; \text{for any} \;\; y^N \in \mathcal{A}_i^N(v) \\ &\geq \; \epsilon - (< y^N - y, a >) \text{ whenever } N > N(\epsilon). \end{aligned}$$

Thus, we need to find $\overline{N}$ such that for all $y \in \mathcal{A}_i(v)$, if $N > \overline{N}$, then

$$< y^N - y, a > < \; \delta$$

for some $y^N \in \mathcal{A}_i^N(v)$. But,

$$< y^N - y, a > \quad \leq \quad \|y^N - y\| \; \|a\|.$$

And, we know that for any $y \in \mathcal{A}_i(v)$ we can find a $y^N \in \mathcal{A}_i^N(v)$ such that $\|y^N - y\| \leq \frac{\delta}{\|a\|}$ whenever $N > \overline{N}(\frac{\delta}{\|a\|})$ by Theorem 6. Then we have that

$$< y, a > \geq \quad \epsilon - \delta \text{ whenever } N > \max\{N(\epsilon), \overline{N}(\frac{\delta}{\|a\|})\}$$

for any $y \in \mathcal{A}_i(v)$. $\diamondsuit$

As a corollary to our previous theorem, we have that an approximated detection signal which is strictly proper for the approximated models is also strictly proper for the real models for sufficiently large N.

Corollary 3. *Suppose $v^N \to v$ and for some N sufficiently large, there is an approximated detection signal v^N such that for all $y^N \in \mathcal{A}_i^N(v^N)$,*

$$< y^N, a > \geq \quad \epsilon \quad \textit{whenever } N > N(\epsilon)$$

where a as any supporting hyperplane. Then for any $\delta > 0$, there exists $\overline{N}$ such that if $n > \overline{N}$ then for all $y \in \mathcal{A}_i(v^N)$,

$$< y, a > \geq \quad \epsilon - \delta.$$

Proof. Let

1. $y^N = y^N(t, x_0^N, \mu_1^N, v^N, \mu_2^N) \in \mathcal{A}_i^N(v^N)$
2. $y = y(t, x_0, \mu_1, v^N, \mu_2) \in \mathcal{A}_i(v^N)$
3. $y^N(v) = y^N(t, x_0^N, \mu_1^N, v, \mu_2^N) \in \mathcal{A}_i^N(v)$
4. $y(v) = y(t, x_0, \mu_1, v, \mu_2) \in \mathcal{A}_i(v)$

Then

$$\begin{aligned} < y, a > &= \; < y^N, a > -(< y^N, a > - < y, a >) \text{ for any } \; y \in \mathcal{A}_i(v^N) \\ &\geq \; \epsilon - (< y^N, a > - < y, a >). \end{aligned}$$

Thus, we need to find $\overline{N}$ such that for all $y^N \in \mathcal{A}_i^N(v^N)$, if $N > \overline{N}$, then

$$< y^N - y, a > \; < \; \delta$$

for some $y \in \mathcal{A}_i(v^N)$.

For convenience, we change variables, using the variables as described in [34] and Section 5.1. Note that this is merely a change in representation of the system, not an approximation of the system. Note also that $\mathcal{M}^N = \mathcal{M}$, $\mathcal{C}^N = \mathcal{C}$, and $\mathcal{B}^N = \mathcal{B}$. Suppose

1. $L_1(z_0) = S(t)z_0$
2. $L_1^N(z_0) = S^N(t)P^N z_0$
3. $L_2(\phi) = \int_0^t S(t-s)\mathcal{M}f(s)ds$
4. $L_2^N(\phi) = \int_0^t S^N(t-s)\mathcal{M}f(s)ds$
5. $L_3(f) = \int_0^t S(t-s)\mathcal{B}f(s)ds$
6. $L_3^N(f) = \int_0^t S^N(t-s)\mathcal{B}f(s)ds$

where S^N is the semigroup on Z^N generated by $\mathcal{A}^{\mathcal{N}}$, i.e., $S^N(t) = e^{\mathcal{A}^N t}$, in the approximated system. Let $a = \begin{bmatrix} a \\ 0 \end{bmatrix}$ so that the dimensions are appropriate for our problem. Then

$$\begin{aligned}
< y - y^N, a > &= < C(z - z^N), a > + < N(\mu_1 - \mu_1^N), a > \\
&= < C[(L_1(z_0) + L_2(\mu) + L_3(v^N)) \\
&\quad -(L_1^N(z_0) + L_2^N(\mu) + L_3^N(v^N))], a > + < N(\mu_1 - \mu_1^N), a > \\
&= < C[(L_1(z_0) + L_2(\mu) + L_3(v^N)) - (L_1^N(z_0) + L_2^N(\mu) + L_3^N(v^N)) \\
&\quad +L_3^N(v) - L_3^N(v) + L_3(v) - L_3(v)], a > + < N(\mu_1 - \mu_1^N), a > \\
&= < C[(L_1(z_0) + L_2(\mu) + L_3(v)) - (L_1^N(z_0) + L_2^N(\mu) + L_3^N(v)) \\
&\quad +L_3^N(v^N) - L_3^N(v) + L_3(v^N) - L_3(v)], a > + < N(\mu_1 - \mu_1^N), a > \\
&= < (y(v) - y^N(v)), a > + < C(L_3^N(v^N - v) + L_3(v^N - v)), a > \\
&= < (y(v) - y^N(v)), a > + \| C(L_3^N(v^N - v) + L_3(v^N - v))\| \|a\|.
\end{aligned}$$

But, we have by assumption that $v^N \to v$ and since we are working on a closed and bounded interval, so there exists an $N_1(\frac{\delta}{2})$ such that for $N > N_1(\frac{\delta}{2})$, $\|C(L_3^N(v^N - v) + L_3(v^N - v))\|\|a\| < \frac{\delta}{2}$. And, from the proof of our previous theorem, we know we can make $< C(y(v) - y^N(v)), a > < \frac{\delta}{2}$ for some $N_2(\frac{\delta}{2})$. Thus, we have

$$\begin{aligned}
< y(v) - y^N(v), a > + < L_3^N(v^N - v) + L_3(v^N - v), a > &< \; < C(y(v) - y^N(v)), a > + \frac{\delta}{2} \\
&< \frac{\delta}{2} + \frac{\delta}{2} \\
&< \delta \text{ for} \\
&\quad \overline{N} > \max\{N_1(\frac{\delta}{2}), N_2(\frac{\delta}{2})\}.
\end{aligned}$$

Then we have that $< y, a > \geq \;\; \epsilon - \delta$ whenever $N > \max\{N_1(\frac{\delta}{2}), N_2(\frac{\delta}{2}), N(\epsilon)\}$. $\diamondsuit$

We have seen from our theoretical work that the spline approximation is more than capable of getting us good approximations of our detection signal. We have shown that for each 'real' output, there is an approximated output nearby. We have also shown that if a signal will separate the real output sets, then it will separate the approximated output sets. We have seen that a hyperplane which strictly separates the real output sets will also strictly separate the approximated output sets. While our theorems are important, they do not address the question of comparing the various methods. We shall look into how the methods compare in our next chapter by examining a few numerical experiments.

Chapter 6

Examples and Analysis

In this chapter, we compare the Method of Steps, the spline approximation and the central difference approximation on some numerical examples. There are several things to keep in mind as we consider the examples. The first is that our objective is not to find good approximations for the states but rather to find good approximations for the detection signal. Also, as the approximation methods increase the dimensions of the computational problem as we increase the spatial grid, our objective is to be able to get good approximations of the detection signal on coarse grids. Finally, it is important to recall that the Method of Steps, while providing a true solution, only works on a small class of problems. Thus, some of our examples serve the purpose of demonstrating the more flexible approximation methods.

While the first few examples are on simple problems, they serve several purposes. The first is to compare all three methods on a single, simple example. By comparing the approximation methods with the true solution given by the Method of Steps, we can visually gauge how well the approximated solutions are doing. Then we compare the approximated methods on a problem with multiple and mixed delays. Finally, we look at a more complex example.

6.1 Steps, Differences and Splines on a Single Delay

We consider the problem

$$\begin{aligned}
\dot{x_0} &= -2x_0 + x_0(t-1) + v + \mu_{0,1} && (6.1a)\\
y &= x_0 + \mu_{0,2} && (6.1b)\\
\dot{x_1} &= -3x_1 + x_1(t-1) + v + \mu_{1,1} && (6.1c)\\
y &= x_1 + \mu_{1,2}. && (6.1d)
\end{aligned}$$

This is an interesting example because we compare the approximated solutions with the

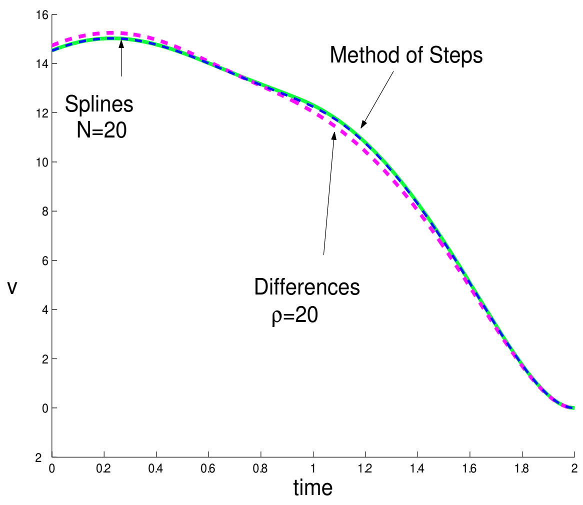

Figure 6.1: The figure shows the Method of Steps, the spline approximation and the difference approximation on the models (7.4) which have a single delay. A mesh size of 20 is used in each approximation.

true solution given by the Method of Steps. We see in Figure 6.1 the detection signal computed using the three methods plotted together, where ρ is the mesh size of the difference approximation and N is the mesh size of the spline approximation. In this example, we use mesh sizes of 20 for both approximations. It is clear from the figure that the approximated methods do very well. It is difficult to distinguish the difference between the various plots. The difference approximation never quite converges to the true solution. It is probably due to a small error in weighting the initial condition. The norms of the three signals are all quite close. The norm of the differences signal is 16.0348. The norm of the spline signal is 16.01667. Finally, the norm of the true signal, given by the Method of Steps, is 16.0434.

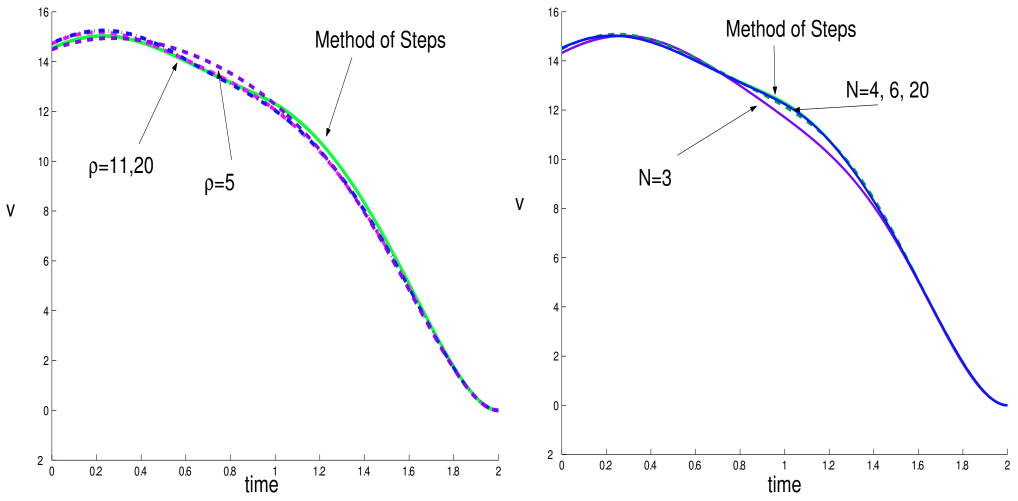

Figure 6.2: The figure compares the Method of Steps and the spline and difference approximations on example (7.4). On the left, we have the auxiliary signal, v, calculated using several meshes and the difference approximation plotted together with the signal determined using the Method of Steps. On the right, we have a similar plot but the signal was approximated using splines.

Figure 6.2 compares the Method of Steps, and the spline and difference approximations. On the left, we have the auxiliary signal, v, calculated using the difference approximation and mesh sizes of 5, 11, and 20. These are plotted together with the signal determined using the Method of Steps. On the right, we have a similar plot but the signal was approximated using splines.

There are several things to notice in these plots. First, one might notice that the spline approximation at mesh size 4 and mesh size 20 are virtually the same. This is not true of the difference approximation. On the other hand, at mesh size 11, the difference approximation is practically indistinguishable from that of mesh size 20. Thus, we see that our objective of using coarse approximations is better met by the spline method but for a large enough ρ, both methods will do the job.

6.2 Splines and Differences on Multiple Delays

Next, we look at the problem

$$\dot{x_0} = -2x_0 + x_0(t-1) + v + \mu_{0,1} \tag{6.2a}$$

$$y = x_0 + \mu_{0,2} \tag{6.2b}$$

$$\dot{x_1} = -3x_1 + x_1(t-.4) + v + \mu_{1,1} \tag{6.2c}$$

$$y = x_1 + \mu_{1,2}. \tag{6.2d}$$

In Figure 6.3, we see the detection signal, v, for example 6.2, which mixed delays. This

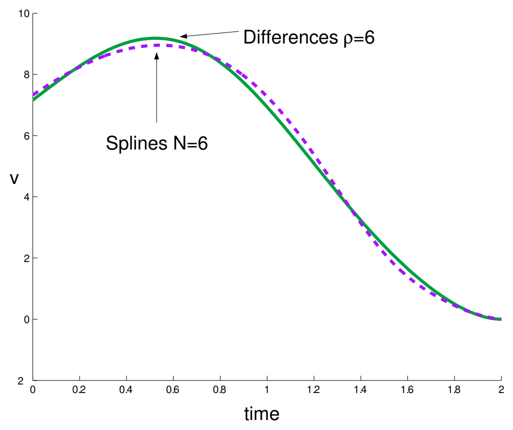

Figure 6.3: The figure shows the detection signal found for example 6.2, which has mixed delays. One of our models has a delay of 1 while the other has a delay of .4. In both cases, we use mesh sizes of 6.

problem is significant in that it would be impossible to solve it with the Method of Steps as there is more than one delay and the delays are not multiples of each other. At a mesh size of only 6 for both approximations, we see the signals are quite similar. The norm for the spline signal found with the spline approximation is 9.1527 while the norm of the signal found with the difference approximation is 9.1436.

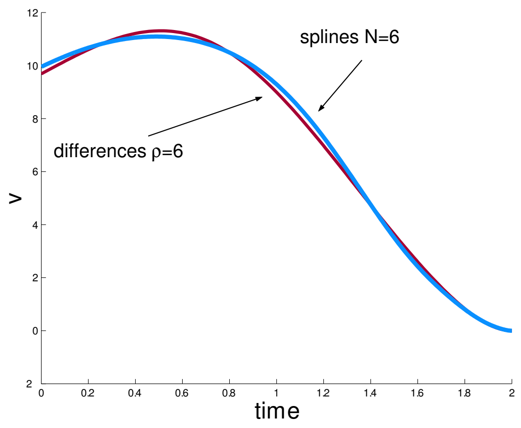

Figure 6.4: Comparison of the spline approximation and the central difference approximation on mixed and multiple delays from example (6.3).

6.3 Splines and Differences on Mixed and Multiple Delays

In this example, we compare the spline approximation and the central difference approximation on multiple delays. We look at the problem

$$\dot{x_0} = -2x_0 + x_0(t-1) + v + \mu_{0,1} \tag{6.3a}$$

$$y = x_0 + \mu_{0,2} \tag{6.3b}$$

$$\dot{x_1} = -3x_1 + .5x_1(t-.4) + .5x_1(t-1) + v + \mu_{1,1} \tag{6.3c}$$

$$y = x_1 + \mu_{1,2} \tag{6.3d}$$

We see in Figure 6.4 the plot of v for example (6.3) with mesh sizes of 6 for both types of approximations. Notice that these models are different from the simpler models of example (6.2). Model 1 has two delays in it which is something many methods are unable to handle. Moreover, Models 0 and 1 have different delays. The detection signals found are fairly close in both approximations. The norm of the differences signal is 11.33 whereas the norm of the splines is 11.72.

6.4 Mach Number in a Wind Tunnel

The next example is a linearized model of the control of the Mach number in a wind tunnel. We adapted these models from Manitus and Tran's *Numerical simulation of a non-linear feedback controller for a wind tunnel model involving a time delay*([48]). Mach number is the speed of an object divided by the speed of sound in the surrounding medium. For example, an aircraft moving twice as fast as the speed of sound is said to be travelling at Mach 2. Model 0 is given by

$$\begin{aligned}
x_1'(t) &= -ax_1(t) + akx_2(t-h) + \nu_1 && (6.4a)\\
x_2'(t) &= x_3(t) + \nu_2 && (6.4b)\\
x_3'(t) &= -\beta^2 x_2(t) - 2\eta\beta x_3(t) + \beta^2(u(t)+v(t)) + \nu_3 && (6.4c)\\
y &= x_1 + \nu_4. && (6.4d)
\end{aligned}$$

where $a = (1.964)^{-1},\ k = -0.0117,\ \beta = 6,\ \eta = 0.8,\ h = 1s$, and $u(t) \equiv 2$. Model 1 is given by

$$\begin{aligned}
x_1'(t) &= -ax_1(t) + akx_2(t-h) + \nu_1 && (6.5a)\\
x_2'(t) &= x_3(t) + \nu_2 && (6.5b)\\
x_3'(t) &= -\beta^2 x_2(t) - 2\eta\beta x_3(t) + \beta^2(u(t)+v(t)) + \nu_3 && (6.5c)\\
y &= x_1 + \nu_4 && (6.5d)
\end{aligned}$$

where $a = (1.2892)^{-1},\ k = -0.0117,\ \beta = 4,\ \eta = 0.8,\ h = 1s$, and $u(t) \equiv 2$. In these models, x_1 is the Mach number, x_2 is the actuator[1] position (guiding vane angle in a driving fan), x_3 is the actuator rate, and v is the input to the actuator servomechanism[2]. The delay, h, is the time of transport between the fan and the test section.

Note that the scaling of variables is set up so that for a constant control, u, we get the set point (equilibrium) of $x_1 = ku$, $x_2 = u$, $x_3 = 0$. We assume that $u(t) = 2$ and that the detection horizon is $[0, \kappa h]$. We will send the auxiliary signal down the same channel as u.

Notice that the failure mode is the same as the normal mode, except that $a = (1.2.892)^{-1}$ and $\beta = 4$ in the failed model. This corresponds to a change in the driving fan. Note that

[1]An actuator is a mechanical device for moving or controlling something

[2]A servomechanism is a self-regulating feedback system

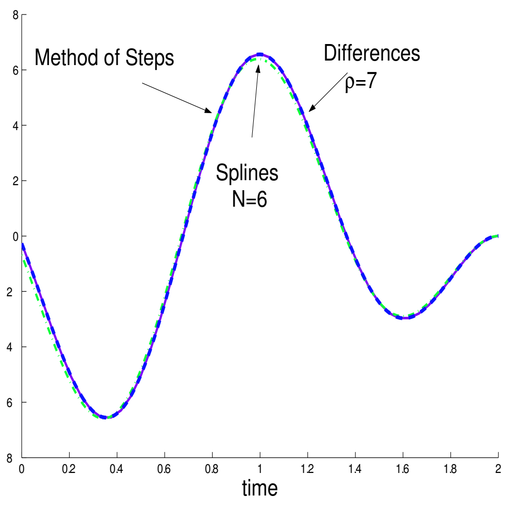

Figure 6.5: This figure represents the detection signals found using splines, differences and the Method of Steps when calculating the signal for a linearized, model of the control of the Mach number in a wind tunnel. A spline approximation of grid size $N = 6$ was used while a difference approximation of grid size $\rho = 7$ was used.

a and β do not appear in the equations defining the set point (equilibrium) so that at set point operation, a change in either of these parameters would not be detectable no matter how long the system was observed. Thus an auxiliary signal is needed to notice changes in the models.

Figure 6.5 shows the detection signal, v, calculated using splines, differences and the Method of Steps for $\kappa = 1$. Notice that both approximations did very well approximating the true detection signal found using the method of steps. We used a fairly coarse grid of $N = 6$ for the spline approximation. For the difference approximation, we used a grid of size $\rho = 7$. The norms are also close, with the norm of the Steps signal being 5.5807, the norm of the spline signal being 5.5398 and the norm of the difference signal being 5.58005. Its worth noting that SOCS had trouble converging on the difference approximation. The solution which we present here is the last solution given before SOCS terminated abnormally. The abnormal termination was caused by a lack of sufficient working storage, technically. However, the reason the storage requirements were so large was due to trying to resolve the dynamics to a relative tolerance of $.100E-06$. As SOCS had resolved the dynamics to a relative tolerance of $0.35E-05$, we accepted this solution as sufficient for our purposes.

6.5 Conclusions about Numerical Examples

Our objectives in working with the approximations were two fold:

1. Good approximations of the detection signal.

2. Work on coarse grids.

The examples suggest that we have achieved our objectives. Unlike the Method of Steps, approximations are able to work on problems with mixed and multiple delays. Moreover, we see that for fairly coarse grids, we are getting good solutions. The difference approximation does not seem to do as well as the spline approximation.

Chapter 7

Future Work and Conclusions

7.1 Future Work

In extending this fault detection and model identification algorithm to problems with delays, we have shown that approximations to the delayed system yield detection signals which will solve the problem, and we have demonstrated interesting theoretical and numerical results on the approximation methods used. However, there remains far more work to be done in this area and below we list just a few potential problems and applications.

Alternative Approximations

We examined only two methods of approximating our delayed system. As mentioned in Section 1.2.1, there are many ways of approximating delayed systems with ODE systems and it would be of benefit to look at some of these other methods. While the results of our spline approximation were very good, perhaps other approximation methods would yield faster, more accurate results.

In [36], Ito and Kappel present formulations of the Trotter-Kato Theorem for approximations of linear C_0-semigroups with the intent that they might be useful for showing convergence of numerical methods for PDE's. In Section 4.1, we reformulate the delayed system into a PDE. This work of Ito and Kappel could be very useful in considering other methods of approximation.

Stochastic Models

The work in this thesis is on deterministic models with uncertainty. However, stochastic models are very common in real applications. A problem of interest would be examining the fault detection and model identification algorithm on stochastic models. While the method should extend naturally to stochastic models, there are some questions to be asked. It is not clear how other information which is generally associated with detection on stochastic models, like error probabilities, is to be derived.

Nonlinear Models

While our work extends the fault detection algorithm to linear models with delays, another useful extension would be to nonlinear models and nonlinear models with delays. Nonlinearities are likely to offer computational difficulties in that having a suitable initial guess is always an issue when looking at nonlinear problems. Also, in general, nonlinearities are likely to produce only locally optimal solutions and cause a loss of convexity in the output sets.

Different types of nonlinearities will raise different issues in problem formulation. In [30], Horton suggests that there are 3 types of nonlinearities which their method should extend to smoothly: small, norm-bounded, nonlinearities in the state, nonlinearities in the control, and coefficient matrices dependent on the control.

Noise

Another important question is that of other types of noise. In this thesis, we examine the question of deterministic models with additive uncertainty, but there are other cases in which model uncertainty is important.

Thus far, we have considered models of the form,

$$\dot{x} = Ax + Bv + M\mu \tag{7.1a}$$

$$y = Cx + N\mu \tag{7.1b}$$

where the noise bound is given by

$$\mathcal{S}(x(0), \mu) = x(0)^T Q x(0) + \int_0^{\omega} |\mu(t)|^2 \ \ dt < 1. \tag{7.2}$$

A generalization of this model is the case where the detection signal, v, acts on the output as well. That is, we have the model

$$\dot{x} = Ax + Bv + M\mu \tag{7.3a}$$

$$y = Cx + Pv + N\mu. \tag{7.3b}$$

Notice that if $P = 0$, then we are back to our previous model.

One way to add model uncertainty to model (7.3) is to adapt an idea from [58] of Savkin and Petersen. We use models of the form

$$\begin{aligned}\dot{x} &= (A + D\Delta G)x + (B + D\Delta H)v + M\mu \\ y &= (C + F\Delta G)x + (P + F\Delta H)v + N\mu\end{aligned}$$

where D, F, G, H are constant matrices and $\|\Delta(t)\| < 1$. Then we can rewrite this system as

$$\dot{x} = Ax + Bv + \begin{bmatrix} D & M \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \tag{7.5a}$$

$$z = Gx + Hv \tag{7.5b}$$

$$y = Cx + Pv + \begin{bmatrix} F & N \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \tag{7.5c}$$

with $\mu_1 = \Delta z,\ \mu_2 = \mu$.

Since $\mu_1 = \Delta z$ then, $|\mu_1(t)| = |\Delta(t)z(t)|$ for all $t \in [0\ \omega]$. Thus, $|\mu_1(t)| \leq |z(t)|$ for all $t \in [0\ \omega]$ as $\|\Delta(t)\| < 1$. This gives us the condition that

$$\int_0^s (|\mu_1|^2 - |z|^2)\ dt < 0 \text{ for all } s \in\ [0, \omega].$$

Thus, our noise measure becomes

$$x(0)^T Q x(0) + \int_0^s (|\mu_1|^2 + |\mu_2|^2 - |z|^2)\ dt < 0 \text{ for all } s \in\ [0, \omega]. \tag{7.6}$$

Of course, there are issues with this formulation. Notice that the problem now has the parameter, s, in it and we have the added constraint (7.5b). Also,the noise measure now has negative terms in it. Our analysis on additive noise models, we have shown the importance of having bounded noise sets in the delayed case. This will remain important in the model noise case.

If $H = 0$ and G is sufficiently small, the constraint (7.5b) could be eliminated by including it directly. However, this will cause the noise bound to be more complex. Moreover, if G is sufficiently small, then the quadratic in (7.6) will be positive definite.

Some research has been conducted on these types of models for the non-delayed case. See [12, 16, 10] for more information.

7.2 Conclusions

We had several goals upon starting this work. First, we wished to determine if it was possible to make the model identification decision early and to devise an algorithm for making that determination when working with ODE models as described in 1.3. We also wished to determine if it was possible to extend the FDMI algorithm to delayed systems by approximating them with ODE systems. In addition, we wished to find an approximation method which would not only find a good detection signal, but one which would do it on coarse grids so as to reduce the computational overhead of solving the problem as much as possible.

In Chapter 1, we reviewed material relevant to our research. First, we looked at some mathematical concepts which were to be important throughout the body of our work. Then we examined the work done by others. In addition to examining prior research done on fault detection, model identification and approximation of delayed systems, we also looked in detail at the FDMI algorithm developed in [30, 14, 52]. This work of Campbell, Nikoukhah, and Horton served as the basis of our work.

Having developed the context for our problem, we went in Chapter 2 to explore the question of whether or not it is possible to make the model identification decision early. We showed that it is possible to make the determination early. We also devised an algorithm which can be used to make the decision. Then we looked at some interesting computational tests and showed that on average, it is possible to reduce the detection time by about 25%.

Next, we explored the question of whether it was possible to apply the FDMI algorithm to problems with delays. Rather than develop a new theory directly in terms of delayed systems, we extended the well-developed theory of [30] to our problems by reformulating the delayed systems. First, we noted in Chapter 3 that it is possible to solve a small class of these problems exactly using the Method of Steps. We detailed a derivation of the necessary conditions for the Method of Steps problem and we proved the optimality of the conditions we found. While the Method of Steps provides a true solution to the problem, its usefulness is limited because of the small class of problems upon which it can be used. However, it served as a basis for comparison for the approximated solutions we developed in Chapter 4 and Chapter 5.

In Chapter 4, we looked at first reformulating the delayed system into a PDE and then approximating the solution to the PDE by central differences. We looked in detail at the reformulation and necessary conditions for the FDMI algorithm. We noted that there is an important addition to the necessary conditions resulting from converting our noise measure to our new variable. That is, we noted that $\phi(0) = 0$.

Then in Chapter 5, we looked at another approximation method. This method, based on the work of Ito and Kappel [34], uses piecewise linear splines to approximate the delayed system by an ODE system. We looked in brief detail at the approximation and formulation of our optimization problem. Then we looked at the problem from a more theoretical point of view, proving that the spline method provides a good approximation of the detection signal by showing that the output sets of the approximated problem were a good approximation of the output sets of the original problem. We also showed that a hyperplane which strictly separates the real output sets strictly separates the approximated output sets. Moreover, we showed that a hyperplane which strictly separates the approximated output sets, strictly separates the real output sets.

Having examined the delayed detection problem analytically and theoretically, in Chapter 6, we looked at some illustrative examples. We first looked at a simple, one dimensional example on models with a single, constant delay. We compared the three methods of solving the delayed problem on this example and observed that our approximations do a good job of finding a detection signal on fairly coarse grids. We also looked at some one dimensional examples which have multiple and mixed delays. These examples are relevant because the Method of Steps, which provided a true solution, will not solve them. We saw in the examples that the detection signals provided by the difference approximation and spline approximation are very similar, both in shape and norm, though the spline approximation seems to arrive at a good signal on a coarser grid than the difference approximation. As grid size has a direct effect on the ease of computation, this is an important difference. Finally, we look at a last, more complex example of the control of Mach Number in a wind tunnel taken from Manitius and Tran [48]. We saw that the spline approximation does very well on this example and the signals are very close for a coarse grid.

Appendix A

Software

A.1 M-files

make_diag.m

The following function makes a block diagonal matrix from a given matrix

```
function new1= matrix_diag(matrix1, sizediag)


%create a blockdiagonal matrix
[n,m]=size(matrix1);
new1=matrix1; for i=2:sizediag
   [n1,m1]=size(new1);

   new1=[new1 zeros(n1,m);
      zeros(n,m1) matrix1];
end
```

A.2 Method of Steps

A.2.1 Matlab Driver for the Method of Steps

```
%system matrices
```

```
%define constants of problem(optional)
mya1=-3;
mya2=-2;

myb1=1
myb2=1
myg1=1;
myg2=1;
myc1=1;
myc2=1;



%define matrices for the original problem
myC1=[myc1];
myC2=[myc2];

myA1=[mya1]
myA2=[mya2]

myB1=[myb1]
myB2=[myb1]

myN1=[0 1 ]
myN2=myN1;

myM1=[1 0  ];

myM2=myM1;


myG1=[myg1]
```

```
myG2=[myg2];

Zero3=zeros(3,3)

disp('eigenvalues of A1')
eig(myA1)



%matrices for the stacked problem.
% note: this is for stacked 2
%change the values of mystacks and the A's
mystacks=2

myAbar1=[myA1 zeros(size(myA1))
   myG1 myA1];
 disp('eigenvalues of ABar1')
eig(myAbar1)

myAbar2=[myA2 zeros(size(myA2))
   myG2 myA2];
   %%%%%%%%%%%%%%%%%%   %%%%%%%%%%%%%%%%%%%%%
   % change nothing below this line
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 disp('eigenvalues of ABar2')
eig(myAbar2)

myBbar1= matrix_diag(myB1,mystacks);
myBbar2= matrix_diag(myB2,mystacks);

myCbar1= matrix_diag(myC1,mystacks);
myCbar2= matrix_diag(myC1,mystacks);
```

```
%___________________________________
myMbar1temp1=matrix_diag(myM1,mystacks);
[myiMbar1temp1,myjMbar1temp1]=size(myMbar1temp1);
[myiG,myjG]=size(myG1);

myMbar1temp2=[myG1
   zeros(myiMbar1temp1-myiG,myjG)];
myMbar1=[myMbar1temp2 myMbar1temp1];
%___________________________________________
myMbar2temp1=matrix_diag(myM2,mystacks);
[myiMbar2temp1,myjMbar2temp1]=size(myMbar2temp1);
[myiG,myjG]=size(myG2);

myMbar2temp2=[myG2
   zeros(myiMbar2temp1-myiG,myjG)];
myMbar2=[myMbar2temp2 myMbar2temp1];
%___________________________________________



myNbar1temp1=matrix_diag(myN1,mystacks);
[myiG,myjG]=size(myG1);
[myiNbar1temp1,myjNbar1temp1]=size(myNbar1temp1);
myNbar1temp2=zeros(myiNbar1temp1,myjG);
myNbar1=[myNbar1temp2 myNbar1temp1];

 %___________________________________________
```

```
myNbar2temp1=matrix_diag(myN2,mystacks);
[myiG,myjG]=size(myG2);
[myiNbar2temp1,myjNbar2temp1]=size(myNbar2temp1);
myNbar2temp2=zeros(myiNbar2temp1,myjG);
myNbar2=[myNbar2temp2 myNbar2temp1];

A0=myAbar1
A1=myAbar2
B0=myBbar1
B1=myBbar2
C0=myCbar1
C1=myCbar2
M0=myMbar1
M1=myMbar2
N0=myNbar1
N1=myNbar2

%may need to do constant orthogonal change of coords on the noise
%do it via a QR decomp on N_i^T to get [Nb_i 0],
%where Nb_i is invertible, also gives [Mb_i Mt_i]
[Q0,R0]=qr(N0')
[Q1,R1]=qr(N1')
pause
%now N_i^T = Q_i * R_i, so N_i = R_i^T * Q_i^T
%and Q_i^T is an orthogonal matrix
%absorb Q_i^T into the noise vector nu_i to get new noise vector
%and Nb_i becomes the invertible part of R_i^T
%may need to fix signs in Q_i and/or R_i
R0=-R0;
R1=-R1;
```

```
Q0(1,1)=-Q0(1,1);
Q0(2,2)=-Q0(2,2);
Q1(1,1)=-Q1(1,1);
Q1(2,2)=-Q1(2,2);
%break down into Mb_i, Mt_i, Nb_i, and Nt_i (zeros)
[mN0,nN0]=size(N0);
[mN1,nN1]=size(N1);
mnN0=min(mN0,nN0);

mnN1=min(mN1,nN1);
mxN0=max(mN0,nN0);
mxN1=max(mN1,nN1);
N0n=R0';
N1n=R1';
Nb0=N0n(1:mnN0,1:mnN0);
Nb1=N1n(1:mnN1,1:mnN1);
Nt0=N0n(:,mnN0+1:mxN0); %just need the size of this
Nt1=N1n(:,mnN1+1:mxN1); %and this
M0n=M0*Q0';
M1n=M1*Q1';
Mb0=M0n(:,1:mnN0);
Mb1=M1n(:,1:mnN1);
Mt0=M0n(:,mnN0+1:mxN0);
Mt1=M1n(:,mnN1+1:mxN1);
%create reduced model system matrices
[mA0,nA0]=size(A0);
[mA1,nA1]=size(A1);
[mMt0,nMt0]=size(Mt0);
[mMt1,nMt1]=size(Mt1);
[mNb,nNb]=size(Nb1);
[mNt0,nNt0]=size(Nt0);
```

```
[mNt1,nNt1]=size(Nt1);
A=[A0-Mb0*inv(Nb0)*C0 Mb0*inv(Nb0)*C1; zeros(mA1,nA0) A1]
M=[Mt0 Mb0*inv(Nb0)*Nb1 zeros(mMt0,nMt1); zeros(mMt1,nMt0) Mb1 Mt1]
B=[B0;B1]
C=[inv(Nb0)*C0 -inv(Nb0)*C1];
N=inv(Nb0)*[zeros(mNb,nNt0) Nb1 zeros(mNb,nNt1)];
%Q and H without beta
Qnb=2*C'*C
Hnb=-4*C'*N
%size of upper left block of R
ulident=nNt0
%Nb1'Nb0'^-1Nb0^-1Nb1 for the center block of R
Nb1
Nmess=Nb1'*inv(Nb0')*inv(Nb0)*Nb1
[nnmess,mnmess]=size(Nmess)
%size of lower right block of R
lrident=nNt1
rtop=[eye(ulident,ulident) zeros(lrident, nnmess+lrident)]
rmiddle= [zeros(nnmess,ulident) Nmess zeros(nnmess,lrident)]
rbottom=[zeros(ulident, nnmess+ulident) eye(lrident,lrident)]
rtogether=[45*rtop;78*rmiddle;99*rbottom]

%end of routine
if(0)
save q.txt Qnb -ascii


save b.txt B -ascii


save c.txt C -ascii


save m.txt M -ascii
save r.txt rtogether -ascii

save n.txt Nmess -ascii

save a.txt A -ascii

save h.txt Hnb -ascii
end
```

A.2.2 Method of Steps Driver

The following program is the driver for the Method of Steps for our first simple example.

```
      PROGRAM setpoint1

c   the big feature of this program is the restricted beta

      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)

      INTEGER NIW,MAXPHS,NW,MAXCS,MAXDP
      PARAMETER (NIW =20050000,MAXPHS = 5,NW = 20005000,MAXCS=1000500)
      PARAMETER (MAXDP = 1000)
      INTEGER IW(NIW),IPCPH(MAXPHS+1),IPDPH(MAXPHS+1),NEEDED,IER
      DOUBLE PRECISION W(NW),CSTAT(MAXCS),DPARM(MAXDP)
      DOUBLE PRECISION h,a1,a2,g1,g2,b1,b2,m1,m2
      DOUBLE PRECISION n2,T0,c1,c2,tau1,tau,n1


      EXTERNAL setpoint1IN,dumyig,setpoint1DE,GETTINGvPF,DUMYPR
      open (18, FILE='darn.dat', STATUS='OLD')
      CALL HHSOCS('ipuocp=18')
      CALL HHSOCS('IPGRD=20')
      CALL HHSOCS('IPODE=20')
      CALL HHSNLP('MAXNFE=100000')
      CALL HHSOCS('mxpcon=100')
      CALL HHSNLP('NITMAX=500')
c      CALL HHSNLP('KTOPTN=SMALL')
c      CALL HHSNLP('tolpvt=.1')
      CALL HHSOCS('MXSTAT=300')
      CALL HHSNLP('IOFLAG=20')
c     CALL HHSNLP('ALGOPT=FM')
      CALL HHSOCS('MITODE=20')
      CALL HHSOCS('ODETOL=1.D-7')
      CALL HHSOCS('SPRTHS=SPARSE')
      CALL HHSOCS('MTSWCH=3')
      CALL HHSOCS('NSSWCH=1')
C      CALL HHSOCS('ITSWCH=2')
c      CALL Hhsnlp('CONTOL=1.D-6')
c     CALL HHSNLP('OdeTOL=1.D-4')
      call hhsocs('mxterm=200')

      write(18,*) '****************************************'
      write(18,*) 'error code', ier
      write(18,*) 'needed', needed
      write(18,*) 'normal termination'

      CALL HDSOCS(setpoint1IN,dumyig,setpoint1DE,GETTINGvPF,DUMYPR,
     +            IW,NIW,W,NW,MAXPHS,CSTAT,MAXCS,IPCPH,DPARM,MAXDP,
     +            IPDPH,NEEDED,IER)
      write(*,*) '****************************************'
      write(*,*) 'error code', ier
      write(*,*) 'needed', needed

      close(18)
       END

      SUBROUTINE setpoint1IN(IPHASE,NPHS,METHOD,NSTG,NCF,NPF,NPV,NAV,
     +                  NGRID,INIT,MAXMIN,MXPARM,P0,PLB,PUB,PLBL,
     +                  MXSTAT,Y0,Y1,YLB,YUB,STSKL,STLBL,MXPCON,CLB,
     +                  CUB,CLBL,MXTERM,COEF,ITERM,TITLE,IER)

      implicit double precision (a-h,o-z)



      INTEGER IPHASE,NPHS,METHOD,NSTG,NCF(3),NPF(2),NPV,NAV,NGRID,
     +        INIT,MAXMIN,MXPARM,MXSTAT,MXPCON,MXTERM,ITERM(4,MXTERM),
     +        IER,NTERM,NPATH, indextracker

      DOUBLE PRECISION P0(MXPARM),PLB(MXPARM),PUB(MXPARM),Y0(0:MXSTAT),
     +          Y1(0:MXSTAT),YLB(-1:1,0:MXSTAT),YUB(-1:1,0:MXSTAT),
     +          STSKL(0:MXSTAT+MXPARM,2),CLB(MXPCON),CUB(MXPCON),
     +          COEF(MXTERM), initconditon, xi

      CHARACTER PLBL(MXPARM+2)*80,STLBL(0:MXSTAT)*80,
     +          CLBL(0:MXPCON)*80,TITLE(3)*60

c      METHOD = 2
c      NSTG = 2
      NCF(1) =  9
      NCF(2)=8
      NAV = 10
      NCF(3)=1
      maxmin=-1
      npv=1
c number of point functions
      npf(1)=3
      npf(2)=0

      write(*,*) ' numstateeq=', ncf(1), ' TFinal= ', TFinal
      write(*,*) 'num algeq=', ncf(2),'numcont=', nav
      NGRID=13
      INIT = 1
c      BIGBND = ONE/HDMCON(5)

C     ----GUESS FOR INITIAL TIME AND BOUNDARY CONDITION
c      ---------the beginning and final times are fixed.
      Y0(0) = 0
      Y1(0) = 1
      YLB(-1,0) = Y0(0)
      YUB(-1,0) = Y0(0)
      YLB(1,0) = Y1(0)
      YUB(1,0) = Y1(0)
      write(*,*) 'y1', y1(0)

C----DEFINE INITIAL guess and bounds CONDITIONS FOR STATE VARIABLES
c---on the first and last interval, we want psi minus and psi plus
c----------set everything to 0
        DO 150 I=1, ncf(1)+nav
        Y0(I) =10
        Y1(I) =50
 150    CONTINUE
```

```
       z1=0
       z2=2
       lam1=4
       lam2=6
       bigz=8
       mu1=9
       mu2=13
       v=17
       beta =0
c labels
      stlbl(0)='time    time'
      stlbl(1)='z11     z11'
      stlbl(2)='z12     z12'
      stlbl(3)='z21     z21'
      stlbl(4)='z22     z22'
      stlbl(5)='lam11   lam1'
      stlbl(6)='lam12   lam12'
      stlbl(7)='lam21   lam21'
      stlbl(8)='lam22   lam22'
      stlbl(9)='bigz    bigz'


c---------SET specific conditions
c-------lamda_0(0)=lamda_kappa(h)=0

c__________this is actually the last lam1
         YLB(1,lam2)= 0
         YUB(1,lam2)= 0
c__________this is actually the last lam2
        YLB(1,bigz)= 0
        YUB(1,bigz)= 0




c-------Z(h)>=1
        YLB(1,bigz+1)= 1

c---------------set the parameter

          p0(1)=.5

c-------.01<=beta<=.99
       PLB(1)= .45
       PUB(1)= .69
       plbL(1)='BETA    BETA'

c****************************************************
c****************************************************
c****************************************************
C-------  Describes inequality  or algebraic constraint
c-------  these constraints are described in F

C  defined an a loop

        NTERM=0
        NPATH=0
c--------define the terms that are the path constraints
      DO 800 I=1,ncf(2)
         NPATH = npath+1
         nterm=nterm+1
c----calculate second point function
```

```
c------------define what quantity term j belongs to

         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in

         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1

         ITERM(3,NTERM) = 0

c--------------define which constraint this will be used in
         ITERM(4,NTERM) = -i
         CLB(i)=0
         cub(i)=0
         COEF(NTERM) = 1.


  800    CONTINUE
c*******************************************************
c*******************************************************
c---THE BOUNDARY CONDITIONS
c-----index tracker keeps track of the number of terms
c-----so that we may go back and use them to collect the
c-----second set
c__________z_1

c-----define point functions for bounds
```

```
c--------this loop does -z_1(tf)
        NTERM = NTERM + 1
        NPATH = NPATH+1





c-------calculate first point function
c--------calculated at the begining of the phase
c------------define what quantity term j belongs to
         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in
         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1
         ITERM(3,NTERM) = 1

c--------------define which constraint this will be used in
c-----------this is the index in the psi matrix
         ITERM(4,NTERM) = z1+1
            COEF(NTERM) = -1.


c-----this loop does  z_1(0)
c-----it goes back to the npath that we started with before
        NTERM = NTERM + 1
c----calculate second point function
c------------define what quantity term j belongs to
         ITERM(1,NTERM) = npath
```

```
c-------------define which phase term j is computed in

         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1

         ITERM(3,NTERM) = -1

c--------------define which constraint this will be used in
c-----we want x1(t0)-x2(tf), x2(t0)-x3(tf).....xn-1(t0)-xn(tf)
c---------the first set of terms (t0s) are in psi(1)..psi(n-1)
c----------the second set of terms (tfs) are in psi(n)...(2n-2)
         ITERM(4,NTERM) = z1+2
         CLB(npath)=0
         cub(npath)=0
         COEF(NTERM) = 1.
c********************************************************
c********************************************************
c---THE BOUNDARY CONDITIONS
c-----index tracker keeps track of the number of terms
c-----so that we may go back and use them to collect the
c-----second set
c__________z_2

c-----define point functions for bounds

c--------this loop does -z_1(tf)
        NTERM = NTERM + 1
        NPATH = NPATH+1
```

```
c-------calculate first point function
c--------calculated at the begining of the phase
c------------define what quantity term j belongs to
         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in
         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1
         ITERM(3,NTERM) = 1

c--------------define which constraint this will be used in
c-----------this is the index in the psi matrix
         ITERM(4,NTERM) = z2+1
            COEF(NTERM) = -1.


c-----this loop does  z_1(0)
c-----it goes back to the npath that we started with before
        NTERM = NTERM + 1
c----calculate second point function
c------------define what quantity term j belongs to
         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in
```

```
         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1

         ITERM(3,NTERM) = -1

c--------------define which constraint this will be used in
c-----we want x1(t0)-x2(tf), x2(t0)-x3(tf).....xn-1(t0)-xn(tf)
c---------the first set of terms (t0s) are in psi(1)..psi(n-1)
c----------the second set of terms (tfs) are in psi(n)...(2n-2)
         ITERM(4,NTERM) = z2+2
         CLB(npath)=0
         cub(npath)=0
         COEF(NTERM) = 1.

C******************************************************
        DO 801 I=2,3
      NPATH = npath+1

         nterm=nterm+1
c----lambda1(0)-.5beta*x1(0)=0
c---this is evaluated in a point function
c------------define what quantity term j belongs to

         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in

         ITERM(2,NTERM) = 1
```

```
c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1

         ITERM(3,NTERM) = -1
c------define which function this is
c------since
         ITERM(4,NTERM) =  -i
         CLB(npath)=0
         cub(npath)=0

         COEF(NTERM) = 1.
  801   continue
C*******************************************************
c---------calculate Z(0)-blah=0

       NPATH = npath+1
         nterm=nterm+1
c----calculate first point function
c------add point function to the cost

c-------first term is for the x(0) term
c------------define what quantity term j belongs to

         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in

         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1
```

```
         ITERM(3,NTERM) = -1
c-------------
         ITERM(4,NTERM) =  -1
         CLB(NPATH)= 0
          cub(NPATH)=0
C******************************************************

c********************************************************
c********************************************************
c********************************************************
c********************************************************
c********************************************************
c---THE BOUNDARY CONDITIONS
c-----index tracker keeps track of the number of terms
c-----so that we may go back and use them to collect the
c-----second set
c__________lam_1

c-----define point functions for bounds

c--------this loop does -lam_1(tf)
        NTERM = NTERM + 1
        NPATH = NPATH+1




c-------calculate first point function
c--------calculated at the begining of the phase
c------------define what quantity term j belongs to
```

```
         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in
         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1
         ITERM(3,NTERM) = 1

c--------------define which constraint this will be used in
c-----------this is the index in the psi matrix
         ITERM(4,NTERM) = lam1+1
            COEF(NTERM) = -1.



c-----this loop does  z_1(0)
c-----it goes back to the npath that we started with before
        NTERM = NTERM + 1
c----calculate second point function
c------------define what quantity term j belongs to
         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in

         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1

         ITERM(3,NTERM) = -1
```

```
c--------------define which constraint this will be used in
c-----we want x1(t0)-x2(tf), x2(t0)-x3(tf).....xn-1(t0)-xn(tf)
c---------the first set of terms (t0s) are in psi(1)..psi(n-1)
c----------the second set of terms (tfs) are in psi(n)...(2n-2)
         ITERM(4,NTERM) = lam1+2
         CLB(npath)=0
         cub(npath)=0
         COEF(NTERM) = 1.


c********************************************************
c********************************************************
c********************************************************
c********************************************************
c********************************************************
c********************************************************
c---THE BOUNDARY CONDITIONS
c-----index tracker keeps track of the number of terms
c-----so that we may go back and use them to collect the
c-----second set
c__________lam_2

c-----define point functions for bounds

c--------this loop does -lam_2(tf)
        NTERM = NTERM + 1
        NPATH = NPATH+1
```

```
c-------calculate first point function
c--------calculated at the begining of the phase
c------------define what quantity term j belongs to
         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in
         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1
         ITERM(3,NTERM) = 1

c--------------define which constraint this will be used in
c-----------this is the index in the psi matrix
         ITERM(4,NTERM) = lam2+1
            COEF(NTERM) = -1.



c-----this loop does  lam2(0)
c-----it goes back to the npath that we started with before
        NTERM = NTERM + 1
c----calculate second point function
c------------define what quantity term j belongs to
         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in


         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1
```

```
         ITERM(3,NTERM) = -1

c--------------define which constraint this will be used in
c-----we want x1(t0)-x2(tf), x2(t0)-x3(tf).....xn-1(t0)-xn(tf)
c---------the first set of terms (t0s) are in psi(1)..psi(n-1)
c----------the second set of terms (tfs) are in psi(n)...(2n-2)
         ITERM(4,NTERM) = lam2+2
         CLB(npath)=0
         cub(npath)=0
         COEF(NTERM) = 1.


c********************************************************



C******************************************************
C  Describes objective quadrature function
      NCF(3) = 1
      NTERM = NTERM + 1
      NPATH = NPATH+1
      ITERM(1,NTERM) = 0
      ITERM(2,NTERM) = 1
      ITERM(3,NTERM) = 0
      ITERM(4,NTERM) = -(ncf(2)+1)
      COEF(NTERM) = 1.


      RETURN
      END
      SUBROUTINE setpoint1DE(IPHASE,T,Z,NZ,P,NP,F,NF,IFERR)

      implicit double precision (a-h,o-z)


      INTEGER IPHASE,NZ,NP,NF,IFERR,IPATH
      DOUBLE PRECISION T,Z(NZ),P(NP),F(NF)

      F(1) = -3*Z(1)+Z(18)-Z(10)-Z(12)
      F(2) = Z(1)-3*Z(2)+Z(19)+Z(11)
      F(3) = -2*Z(3)+Z(18)-Z(15)-Z(17)
      F(4) = Z(3)-2*Z(4)+Z(19)+Z(16)
      F(5) = -2*P(1)*Z(1)+2*P(1)*Z(3)+2*P(1)*Z(13)+3*Z(5)-Z(6)
      F(6) = -2*P(1)*Z(2)+2*P(1)*Z(4)+2*P(1)*Z(14)+3*Z(6)
      F(7) = 2*P(1)*Z(1)-2*P(1)*Z(3)-2*P(1)*Z(13)+2*Z(7)-Z(8)
      F(8) = 2*P(1)*Z(2)-2*P(1)*Z(4)-2*P(1)*Z(14)+2*Z(8)
      F(9) = (2*P(1)*Z(1)-2*P(1)*Z(3))*Z(1)/2+(2*P(1)*Z(2)-2*P(1)*Z(4))*
     #Z(2)/2+(-2*P(1)*Z(1)+2*P(1)*Z(3))*Z(3)/2+(-2*P(1)*Z(2)+2*P(1)*Z(4)
     #)*Z(4)/2+(-4*P(1)*Z(1)+4*P(1)*Z(3))*Z(13)/2+(-4*P(1)*Z(2)+4*P(1)*Z
     #(4))*Z(14)/2+Z(10)**2*P(1)+Z(11)**2*P(1)+Z(12)**2*P(1)+Z(13)**2+Z(
     #14)**2+Z(15)**2*(2-2*P(1))/2+Z(16)**2*(2-2*P(1))/2+Z(17)**2*(2-2*P
     #(1))/2
      F(10) = 2*Z(10)*P(1)-Z(5)
      F(11) = 2*Z(11)*P(1)+Z(6)
      F(12) = 2*Z(12)*P(1)-Z(5)
      F(13) = 2*Z(13)-2*P(1)*Z(1)+2*P(1)*Z(3)
      F(14) = 2*Z(14)-2*P(1)*Z(2)+2*P(1)*Z(4)
      F(15) = Z(15)*(2-2*P(1))-Z(7)
      F(16) = Z(16)*(2-2*P(1))+Z(8)
      F(17) = Z(17)*(2-2*P(1))-Z(7)
      F(18) = abs(Z(18))**2+abs(Z(19))**2

      RETURN
      END

       SUBROUTINE GETTINGvPF(IPHASE,IPHEND,T,Z,NZ,P,NP,PSI,NPSI,IFERR)

       INTEGER  IPHASE,IPHEND,NZ,NP,NPSI,IFERR
       double precision T,Z(NZ),P(NP),PSI(NPSI)
       double precision t0

       double precision Q1,Q2

       double precision lamconst

     lamconst=1
     Q1=1

c---beta x_0(0)^T P x_0(0) + (1-beta) x_1(0)^T P x_1(0)

      if (iphend.eq.-1) then
      t0 = 2*Z(1)**2*P(1)*Q1+2*Z(3)**2*(1-P(1))*Q1
      psi(1) = .5*t0-Z(9)
      psi(2) = Z(7)+2*lamconst*Q1*(1-P(1))*Z(3)
      psi(3) = Z(5)+2*lamconst*Q1*P(1)*Z(1)
      endif
      end
```

A.3 Splines

A.3.1 Matlab Spline Driver

The following Matlab script generates the system matrices for the spline approximation.

```
%system matrices
%define constants of problem(optional)
%myHat(t1,t2,t3,N,r,theta)

%system matrices
%define constants of problem(optional)
mya=1/1.964;
myk=-.0117
mybeta1=6
mybeta2=4
myn=.8

%define matrices for the original problem
myC1=[1 1 0]
myC2=myC1;

myA1=[-mya 0 0
   0 0 1
   0 -mybeta1^2 -2*myn*mybeta1]
myA2=[-mya 0 0
      0 0 1
   0 -mybeta2^2 -2*myn*mybeta2]

myB1=[0;0;mybeta1^2 ]
myB2=[0;0;mybeta2^2 ]

myN1=[0 0 0 1]
```

```
myN2=myN1;

myM1=[1 0 0 0
      0 1 0 0
      0 0 1 0];

myM2=myM1;



myG1=[0 mya*myk 0
   0 0 0
   0 0 0]
myG2=myG1;

%%%%---------Problem vars---------
%mesh size
N2=5;

%system size
n=3; %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%abs(delay)
r=1;
%identity which is system size
eyen=eye(n);
algA1=myA1;
algA2=myA2;
zerosn=zeros(n);
makehalf=[.5*eyen .5*eyen]
%note..we go to N+2 because 0,,N+1
for i=1:(N2+1)
```

```
   t(i)=-(i-1)/N2
end

%%%%-----estimate the hat functions---------%%%%

%%%Create Q
 algQ=[ eyen     zerosn    zerosn    zerosn     zerosn
        makehalf zerosn    zerosn    zerosn     zerosn
        zerosn   makehalf  zerosn    zerosn     zerosn
        zerosn   zerosn    makehalf  zerosn     zerosn
        zerosn   zerosn    zerosn    makehalf    zerosn
        zerosn   zerosn    zerosn    zerosn     makehalf];
 %--note...the 0=t_-1=t_0 and -r=t_N=t_N+1
 %--t=-1,0,1
 algD0= myA1*myHat(t(2),t(1),t(1),N2,r,0)+myG1*myHat(t(2),t(1),t(1),N2,r,-r)
 %--t=0,1,2
 algD1= myA1*myHat(t(3),t(2),t(1),N2,r,0)+myG1*myHat(t(3),t(2),t(1),N2,r,-r)
 %--t=1,2,3
 algD2= myA1*myHat(t(4),t(3),t(2), N2,r,0)+myG1*myHat(t(4),t(3),t(2),N2,r,-r)
 algD3= myA1*myHat(t(5),t(4),t(3), N2,r,0)+myG1*myHat(t(5),t(4),t(3) ,N2,r,-r)
 algD4= myA1*myHat(t(6),t(5),t(4), N2,r,0)+myG1*myHat(t(6),t(5),t(4) ,N2,r,-r)
 algD5= myA1*myHat(t(6),t(6),t(5), N2,r,0)+myG1*myHat(t(6),t(6),t(5) ,N2,r,-r)

algH1=[algD0 algD1 algD2 algD3 algD4 algD5
   zerosn zerosn zerosn zerosn zerosn zerosn
   zerosn zerosn zerosn zerosn zerosn zerosn
   zerosn zerosn zerosn zerosn zerosn zerosn
   zerosn zerosn zerosn zerosn zerosn zerosn
   zerosn zerosn zerosn zerosn zerosn zerosn];

   nreye=(N2/r)*eyen;
   mnreye=(-N2/r)*eyen;

algH2=[zerosn zerosn zerosn zerosn  zerosn zerosn
       nreye  mnreye zerosn zerosn  zerosn zerosn
       zerosn nreye  mnreye zerosn  zerosn zerosn
       zerosn zerosn nreye  mnreye  zerosn zerosn
       zerosn zerosn zerosn nreye   mnreye zerosn
       zerosn zerosn zerosn zerosn  nreye  mnreye];

algH=algH1+algH2;
   algB=zeros(((n*(N2+1))-n),1);

  myB1=[myB1
       algB];

     t1=-1:1/100:0;

     for i=1:length(t1)
       plot(t1(i),myHat(t(6),t(6),t(5), N2,r,t1(i)),'.-')
        hold on
     end

   algM0=zeros(((n*(N2+1))-n),4);

   myM1=[myM1
      algM0];

algC=zeros(1,((n*(N2+1))-n));
myC1=[myC1  algC];
```

```
myA1=algH*inv(algQ)
%_______________________________________
 %--note...the 0=t_-1=t_0 and -r=t_N=t_N+1
 %--t=-1,0,1
 algD0= myA2*myHat(t(2),t(1),t(1),N2,r,0)+myG2*myHat(t(2),t(1),t(1),N2,r,-r)
 %--t=0,1,2
 algD1= myA2*myHat(t(3),t(2),t(1),N2,r,0)+myG2*myHat(t(3),t(2),t(1),N2,r,-r)
 %--t=1,2,3
 algD2= myA2*myHat(t(4),t(3),t(2), N2,r,0)+myG2*myHat(t(4),t(3),t(2) ,N2,r,-r)
 algD3= myA2*myHat(t(5),t(4),t(3), N2,r,0)+myG2*myHat(t(5),t(4),t(3) ,N2,r,-r)
 algD4= myA2*myHat(t(6),t(5),t(4), N2,r,0)+myG2*myHat(t(6),t(5),t(4) ,N2,r,-r)
 algD5= myA2*myHat(t(6),t(6),t(5), N2,r,0)+myG2*myHat(t(6),t(6),t(5) ,N2,r,-r)

algH1=[algD0 algD1 algD2 algD3 algD4 algD5
   zerosn zerosn zerosn zerosn zerosn zerosn
   zerosn zerosn zerosn zerosn zerosn zerosn
   zerosn zerosn zerosn zerosn zerosn zerosn
   zerosn zerosn zerosn zerosn zerosn zerosn
   zerosn zerosn zerosn zerosn zerosn zerosn];

   nreye=(N2/r)*eyen;
   mnreye=(-N2/r)*eyen;

algH2=[zerosn zerosn zerosn zerosn  zerosn zerosn
       nreye  mnreye zerosn zerosn  zerosn zerosn
       zerosn nreye  mnreye zerosn  zerosn zerosn
       zerosn zerosn nreye  mnreye  zerosn zerosn
       zerosn zerosn zerosn nreye   mnreye zerosn
       zerosn zerosn zerosn zerosn  nreye  mnreye];

algH=algH1+algH2;
```

```
   algB=zeros(((n*(N2+1))-n),1);

  myB2=[myB2

    algB];

%   t1=-1:1/100:0

%   for i=1:length(t1)
%      plot(t1(i),myE(t(2),t(3) ,t1(i)),'.-')
%      hold on
%   end

   algM0=zeros(((n*(N2+1))-n),4);

  algM1=[1 0 0 0
  0 0 1 0];

    myM2=[myM2
      algM0];
algN=[0 1 ];

algC=zeros(1,((n*(N2+1))-n));
myC2=[myC2  algC];


myA2=algH*inv(algQ)

A0= myA1
A1= myA2
B0=myB1
```

```
B1=myB2
C0=myC1
C1=myC2
M0=myM1
M1=myM2
N0=myN1
N1=myN2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% From Kirk's thesis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%may need to do constant orthogonal change of coords on the noise
%do it via a QR decomp on N_i^T to get [Nb_i 0],
%where Nb_i is invertible, also gives [Mb_i Mt_i]
[Q0,R0]=qr(N0')
[Q1,R1]=qr(N1')
pause
%now N_i^T = Q_i * R_i, so N_i = R_i^T * Q_i^T
%and Q_i^T is an orthogonal matrix
%absorb Q_i^T into the noise vector nu_i to get new noise vector
%and Nb_i becomes the invertible part of R_i^T
%may need to fix signs in Q_i and/or R_i
R0=-R0;
R1=-R1;
Q0(1,1)=-Q0(1,1);
Q0(2,2)=-Q0(2,2);
Q1(1,1)=-Q1(1,1);
Q1(2,2)=-Q1(2,2);
%break down into Mb_i, Mt_i, Nb_i, and Nt_i (zeros)
[mN0,nN0]=size(N0);
[mN1,nN1]=size(N1);
mnN0=min(mN0,nN0);
```

```
mnN1=min(mN1,nN1);
mxN0=max(mN0,nN0);
mxN1=max(mN1,nN1);
N0n=R0';
N1n=R1';
Nb0=N0n(1:mnN0,1:mnN0);
Nb1=N1n(1:mnN1,1:mnN1);
Nt0=N0n(:,mnN0+1:mxN0); %just need the size of this
Nt1=N1n(:,mnN1+1:mxN1); %and this
M0n=M0*Q0';
M1n=M1*Q1';
Mb0=M0n(:,1:mnN0);
Mb1=M1n(:,1:mnN1);
Mt0=M0n(:,mnN0+1:mxN0);
Mt1=M1n(:,mnN1+1:mxN1);
%create reduced model system matrices
[mA0,nA0]=size(A0);
[mA1,nA1]=size(A1);
[mMt0,nMt0]=size(Mt0);
[mMt1,nMt1]=size(Mt1);
[mNb,nNb]=size(Nb1);
[mNt0,nNt0]=size(Nt0);
[mNt1,nNt1]=size(Nt1);
A=[A0-Mb0*inv(Nb0)*C0 Mb0*inv(Nb0)*C1; zeros(mA1,nA0) A1]
M=[Mt0 Mb0*inv(Nb0)*Nb1 zeros(mMt0,nMt1); zeros(mMt1,nMt0) Mb1 Mt1]
B=[B0;B1]
C=[inv(Nb0)*C0 -inv(Nb0)*C1];
N=inv(Nb0)*[zeros(mNb,nNt0) Nb1 zeros(mNb,nNt1)];
%Q and H without beta
Qnb=2*C'*C
```

```
Hnb=-4*C'*N
%size of upper left block of R
ulident=nNt0
%Nb1'Nb0'^-1Nb0^-1Nb1 for the center block of R
Nb1
Nmess=Nb1'*inv(Nb0')*inv(Nb0)*Nb1
[nnmess,mnmess]=size(Nmess)
%size of lower right block of R
lrident=nNt1
rtop=[eye(ulident,ulident) zeros(lrident, nnmess+lrident)]
rmiddle= [zeros(nnmess,ulident) Nmess zeros(nnmess,lrident)]
rbottom=[zeros(ulident, nnmess+ulident) eye(lrident,lrident)]
disp('the constants in Rtogether are for easy search and replacing')
rtogether=[45*rtop;78*rmiddle;99*rbottom]

%end of routine
if(1)
save q.txt Qnb -ascii

save b.txt B -ascii

save c.txt C -ascii

save m.txt M -ascii

save r.txt rtogether -ascii

save n.txt Nmess -ascii

save a.txt A -ascii
save h.txt Hnb -ascii
end
```

myHat.m

```
function new1= myHat(tkplus1,tk,tkminus1,N,r,theta)

if(tkplus1==tk)
    if((tk <= theta) & (theta <=tkminus1))

       new1=(N/r)*(tkminus1-theta);
      disp('2')
    elseif((tkplus1 <= theta) & (theta <=tk))
   new1=(N/r)*(theta-tkplus1);
   disp('1')

    else
       new1=0;
          disp('3')

      end
   else
        if((tkplus1 <= theta) & (theta <=tk))
          new1=(N/r)*(theta-tkplus1);
           disp('1')
        elseif((tk <= theta) & (theta <=tkminus1))

          new1=(N/r)*(tkminus1-theta);
          disp('2')

        else
```

```
          new1=0;
         disp('3')

     end
end
```

A.3.2 Spline Driver

The following program is a driver for SOCS which solves our first example using the spline approximation.

```
        PROGRAM splineapprox

      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)

      INTEGER NIW,MAXPHS,NW,MAXCS,MAXDP
      PARAMETER (NIW =20050000,MAXPHS = 5,NW = 20005000,MAXCS=1000500)
      PARAMETER (MAXDP = 1000)
      INTEGER IW(NIW),IPCPH(MAXPHS+1),IPDPH(MAXPHS+1),NEEDED,IER
       DOUBLE PRECISION W(NW),CSTAT(MAXCS),DPARM(MAXDP)

      EXTERNAL setpoint1IN,dumyig,setpoint1DE,GETTINGvPF,DUMYPR
       call hhsocs('default')
        call hhsnlp('sparse default')
c        write(*,*) 'running'
         open (55, FILE='prog3.dat', STATUS='OLD')
         CALL HHSOCS('ipuocp=55')
c      CALL HHSOCS('socout=i9j9k9')
      CALL HHSOCS('IPODE=20')
      CALL HHSOCS('IPNLP=20')
      call hhsocs('ipgrd=20')
c       CALL HHSOCS('socout=i9a9b9c9d9e9f9g9h9j9k9l9m9n9o9p9q9r9')
```

```
      CALL HHSNLP('MAXNFE=5000000')
      CALL HHSOCS('mxpcon=100')
      CALL HHSNLP('NITMAX=500')
c      CALL HHSNLP('KTOPTN=SMALL')
c      CALL HHSNLP('tolpvt=.1')
      CALL HHSOCS('MXSTAT=300')
      CALL HHSNLP('IOFLAG=20')
c     CALL HHSNLP('ALGOPT=FM')
      CALL HHSOCS('MITODE=20')
      CALL HHSOCS('ODETOL=1.D-7')
      CALL HHSOCS('SPRTHS=SPARSE')
      CALL HHSOCS('MTSWCH=3')
      CALL HHSOCS('NSSWCH=1')
C      CALL HHSOCS('ITSWCH=2')
c      CALL Hhsnlp('CONTOL=1.D-6')
c     CALL HHSNLP('OdeTOL=1.D-4')
      call hhsocs('mxterm=200')
c-------define parameters----------------
c we have the problem x'=ax(i)+bu(i)+gx(i-1)..kinda...see prob notes



      T0=0



      write(55,*) '****************************************'
      write(55,*) 'main program'
      write(55,*) ' T0=', T0, ' TFinal= ', TFinal,'Findex=',Findex
      write(55,*) '****************************************'

      write(*,*) '****************************************'
      write(*,*) 'main prgram1'
      write(*,*) '***************************************'




      CALL HDSOCS(setpoint1IN,dumyig,setpoint1DE,GETTINGvPF,DUMYPR,
     +            IW,NIW,W,NW,MAXPHS,CSTAT,MAXCS,IPCPH,DPARM,MAXDP,
     +            IPDPH,NEEDED,IER)
      write(*,*) '***************************************'
      write(*,*) 'main prgram '
      write(*,*) 'error code', ier
      write(*,*) 'needed', needed
      write(*,*) '***************************************'


      close(55)


       END
C************************************************************
c************************************************************
      SUBROUTINE setpoint1IN(IPHASE,NPHS,METHOD,NSTG,NCF,NPF,NPV,NAV,
     +                  NGRID,INIT,MAXMIN,MXPARM,P0,PLB,PUB,PLBL,
     +                  MXSTAT,Y0,Y1,YLB,YUB,STSKL,STLBL,MXPCON,CLB,
     +                  CUB,CLBL,MXTERM,COEF,ITERM,TITLE,IER)

      implicit double precision (a-h,o-z)
      INTEGER IPHASE,NPHS,METHOD,NSTG,NCF(3),NPF(2),NPV,NAV,NGRID,
     +        INIT,MAXMIN,MXPARM,MXSTAT,MXPCON,MXTERM,ITERM(4,MXTERM),
     +        IER,NTERM,NPATH

      DOUBLE PRECISION P0(MXPARM),PLB(MXPARM),PUB(MXPARM),Y0(0:MXSTAT),
     +          Y1(0:MXSTAT),YLB(-1:1,0:MXSTAT),YUB(-1:1,0:MXSTAT),
     +          STSKL(0:MXSTAT+MXPARM,2),CLB(MXPCON),CUB(MXPCON),
     +          COEF(MXTERM)

      CHARACTER PLBL(MXPARM+2)*80,STLBL(0:MXSTAT)*80,
     +          CLBL(0:MXPCON)*80,TITLE(3)*60

       integer z1,z2,mu1 ,lam1,lam2,bigz,v,eta

      write(*,*) '***************************************'
      write(*,*) 'initialization'
      write(*,*) '***************************************'

       z1=0
       z2=5
       lam1=10
       lam2=15
       bigz=20
       mu1=21
       v=24
       write(*,*) 'z1=', z1,' z2=', z2
       write(*,*) 'lam1=', lam1,' lam2=', lam2
       write(*,*) 'bigz=', bigz,' mu1=', mu1
       write(*,*) ' v=', v

C  Number of differential equations
```

```
      NCF(1) =  21

c  Number of algebraic equations
      NCF(2)=3

c Number of quadrature equations
      NCF(3)=1

c  Number of algebraic variables
      NAV = 4

c Optimization flag -1..min  0..fease...1..max
      maxmin=-1

c  Number of parameters
      npv=1

c Initial Grid size
      NGRID=23

c type of initial guess
      INIT = 1
c number of point functions
      npf(1)=11
      npf(2)=0

C     ----GUESS FOR INITIAL TIME AND BOUNDARY CONDITION
c      ---------the beginning and final times are fixed.
c

      Y0(0) = 0
      Y1(0) = 2
      YLB(-1,0) = Y0(0)
      YUB(-1,0) = Y0(0)
      YLB(1,0) = Y1(0)
      YUB(1,0) = Y1(0)
      write(*,*) 'y1', y1(0)

C----DEFINE INITIAL guess and bounds CONDITIONS FOR STATE VARIABLES
c---on the first and last interval, we want psi minus and psi plus
c----------set everything to 0
c
        DO 150 I=1, ncf(1)+nav
        Y0(I) =30
        Y1(I) =50
 150    CONTINUE

c--------lamdas
c-------set lam(omega)=0
        DO 152 I=1, z2
        YLB( 1,lam1+i)= 0
        YUB( 1,lam1+i)= 0
c

        YLB( 1,lam2+i)= 0
        YUB( 1,lam2+i)= 0



 152    CONTINUE

c-------Z(h)>=1
        YLB(1,bigz+1)= 1
```

```
c---------------set the parameter

          p0(1)=.5

c-------.01<=beta<=.99
       PLB(1)= .41
       PUB(1)= .69
       plbL(1)='BETA    BETA'

c*****************************************************
c*****************************************************
c*****************************************************
C-------  Describes inequality  or algebraic constraint
c-------  these constraints are described in F

C  defined an a loop

        NTERM=0
        NPATH=0
c--------define the terms that are the path constraints
      DO 800 I=1,ncf(2)
         NPATH = npath+1
         nterm=nterm+1

c------------define what quantity term j belongs to

         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in
```

```
         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1

         ITERM(3,NTERM) = 0

c--------------define which constraint this will be used in

         ITERM(4,NTERM) = -i
         CLB(i)=0
         cub(i)=0
         COEF(NTERM) = 1.


  800    CONTINUE

c*******************************************************
C*****************************************************
        DO 801 I=1,npf(1)
      NPATH = npath+1

         nterm=nterm+1
c----lambda1(0)-.5beta*x1(0)=0
c---this is evaluated in a point function
c------------define what quantity term j belongs to

         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in
```

```
         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1

         ITERM(3,NTERM) = -1
c------define which function this is
c------since
         ITERM(4,NTERM) =  -i
         CLB(npath)=0
         cub(npath)=0

         COEF(NTERM) = 1.
  801   continue

C******************************************************
C  Describes objective quadrature function
      NCF(3) = 1
      NTERM = NTERM + 1
      NPATH = NPATH+1
       ITERM(1,NTERM) = 0
       ITERM(2,NTERM) = 1
       ITERM(3,NTERM) = 0
       ITERM(4,NTERM) = -(ncf(2)+1)
       COEF(NTERM) = 1.


      RETURN
      END
```

```
 SUBROUTINE setpoint1DE(IPHASE,T,Z,NZ,P,NP,F,NF,IFERR)

 implicit double precision (a-h,o-z)



 INTEGER IPHASE,NZ,NP,NF,IFERR
  double precision E1,E2,E3, myE

 DOUBLE PRECISION T,Z(NZ),P(NP),F(NF)

 F(1) = -2*Z(1)-2*Z(2)+2*Z(3)-2*Z(4)+2*Z(5)+Z(25)-Z(22)
 F(2) = 8*Z(1)-8*Z(2)
 F(3) = -8*Z(1)+16*Z(2)-8*Z(3)
 F(4) = 8*Z(1)-16*Z(2)+16*Z(3)-8*Z(4)
 F(5) = -8*Z(1)+16*Z(2)-16*Z(3)+16*Z(4)-8*Z(5)
 F(6) = -Z(6)-2*Z(7)+2*Z(8)-2*Z(9)+2*Z(10)+Z(25)-Z(24)
 F(7) = 8*Z(6)-8*Z(7)
 F(8) = -8*Z(6)+16*Z(7)-8*Z(8)
 F(9) = 8*Z(6)-16*Z(7)+16*Z(8)-8*Z(9)
 F(10) = -8*Z(6)+16*Z(7)-16*Z(8)+16*Z(9)-8*Z(10)
 F(11) = -2*P(1)*Z(1)+2*P(1)*Z(6)+2*P(1)*Z(23)+2*Z(11)-8*Z(12)+8*Z(
#13)-8*Z(14)+8*Z(15)
 F(12) = 2*Z(11)+8*Z(12)-16*Z(13)+16*Z(14)-16*Z(15)
 F(13) = -2*Z(11)+8*Z(13)-16*Z(14)+16*Z(15)
 F(14) = 2*Z(11)+8*Z(14)-16*Z(15)
 F(15) = -2*Z(11)+8*Z(15)
 F(16) = 2*P(1)*Z(1)-2*P(1)*Z(6)-2*P(1)*Z(23)+Z(16)-8*Z(17)+8*Z(18)
#-8*Z(19)+8*Z(20)
 F(17) = 2*Z(16)+8*Z(17)-16*Z(18)+16*Z(19)-16*Z(20)
 F(18) = -2*Z(16)+8*Z(18)-16*Z(19)+16*Z(20)
```

```
      F(19) = 2*Z(16)+8*Z(19)-16*Z(20)
      F(20) = -2*Z(16)+8*Z(20)
      F(21) = (2*P(1)*Z(1)-2*P(1)*Z(6))*Z(1)/2+(-2*P(1)*Z(1)+2*P(1)*Z(6)
     #)*Z(6)/2+(-4*P(1)*Z(1)+4*P(1)*Z(6))*Z(23)/2+Z(22)**2*P(1)+Z(23)**2
     #+Z(24)**2*(2-2*P(1))/2
      F(22) = 2*Z(22)*P(1)-Z(11)
      F(23) = 2*Z(23)-2*P(1)*Z(1)+2*P(1)*Z(6)
      F(24) = Z(24)*(2-2*P(1))-Z(16)
      F(25) = abs(Z(25))**2
      RETURN
      END

       SUBROUTINE GETTINGvPF(IPHASE,IPHEND,T,Z,NZ,P,NP,PSI,NPSI,IFERR)

       INTEGER  IPHASE,IPHEND,NZ,NP,NPSI,IFERR
       double precision T,Z(NZ),P(NP),PSI(NPSI)
       double precision lamconst
       double precision gamma1,gamma2
       double precision Q1,Q2

       Q1=1
       Q2=1
       lamconst=1
       gamma1 = .5*.25
       gamma2 = .5*.25


c---beta x_0(0)^T P x_0(0) + (1-beta) x_1(0)^T P x_1(0)

      if (iphend.eq.-1) then
      psi(1) = Z(11)+2*lamconst*Z(1)*Q1*P(1)
      psi(2) = Z(12)+2*lamconst*Z(2)*gamma1*P(1)
      psi(3) = Z(13)+2*lamconst*Z(3)*gamma1*P(1)
      psi(4) = Z(14)+2*lamconst*Z(4)*gamma1*P(1)
      psi(5) = Z(15)+2*lamconst*Z(5)*gamma2*P(1)
      psi(6) = Z(16)+2*lamconst*Z(6)*Q2*(1-P(1))
      psi(7) = Z(17)+2*lamconst*Z(7)*gamma1*(1-P(1))
      psi(8) = Z(18)+2*lamconst*Z(8)*gamma1*(1-P(1))
      psi(9) = Z(19)+2*lamconst*Z(9)*gamma1*(1-P(1))
      psi(10) = Z(20)+2*lamconst*Z(10)*gamma2*(1-P(1))

      psi(11)=.5*(2*Z(1)**2*Q1*P(1)+2*Z(2)**2*gamma1*P(1)+2*Z(3)**2*gamma1
     #*P(1)+2*Z(4)**2*gamma1*P(1)+2*Z(5)**2*gamma2*P(1)+2*Z(6)**2*Q2
     #*(1-P(1))+2*Z(7)**2*gamma1*(1-P(1))+2*Z(8)**2*gamma1*(1-P(1))+2*
     #Z(9)**2*gamma1*(1-P(1))+2*Z(10)**2*gamma2*(1-P(1)))-Z(21)

      endif
      end
```

A.4 Differences

A.4.1 Matlab to Generate System Matrices

```
%system matrices
%define constants of problem(optional)
%system matrices
%system matrices
%define constants of problem(optional)
%myHat(t1,t2,t3,N,r,theta)
 %INPUT THE VALUE OF RHO
rho=4;
```

```
%system matrices
%define constants of problem(optional)
mya1=-3;
mya2=-2;

myb1=1
myb2=1
myg1=1;
myg2=1;
myc1=1;
myc2=1;



%define matrices for the original problem
myC1=[myc1];
myC2=[myc2];

myA1=[mya1]
myA2=[mya2]

myB1=[myb1]
myB2=[myb1]

myN1=[0 1 ]
myN2=myN1;

myM1=[1 0  ];

myM2=myM1;
```

```
myG1=[myg1]
myG2=[myg2];




s=-1:1/(rho ):0;
sysize=1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CALCULATE THE DELTAS, EPSILONS, AND ALPHAS

%CALCULATE THE FIRST SET OF ALPHAS
epsilon(1)=s(3)-s(2);
delta(1)=s(2)-s(1);

 alpha(1,2)= (delta(1)+epsilon(1))/(delta(1)*epsilon(1)) ;
 alpha(1,3)=-(delta(1)/(epsilon(1)*(epsilon(1)+delta(1))));
 alpha(1,1)=-alpha(1,3)-alpha(1,2);

%CALCULATE THE REST OF THE ALPHAS

for i=2:(rho  )
   for j=2:(rho+1 )
      delta=s(i+1)-s(i);
      epsilon=s(i)-s(i-1);

      alpha(i,1)=-(delta /(epsilon *(epsilon +delta )))
      alpha(i,2)=(epsilon /(delta *(epsilon +delta )))-(delta /(epsilon *(epsilon +de
      if(alpha(i,2)<1^(-14))
         alpha(i,2)=0
         end
      alpha(i,3)=(epsilon /(delta *(epsilon+delta )))

   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CREATE SOME UTILITY MATRICES FOR
%ENABLING VARIABLE SYSTEM SIZES
ident3=eye(sysize);
eye3=eye(sysize);
zeros3=zeros(sysize);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CREATE THE A MATRIX

% the first row is unique
        myA=[alpha(1,1)*eye3   alpha(1,2)*eye3     alpha(1,3)*eye3
             alpha(2,1)*eye3     alpha(2,2)*eye3     alpha(2,3)*eye3];
        [n1,m1]=size(myA);

% all the rest of the rows
      for i=3:(rho )
         [n1,m1]=size(myA);
            myA=[myA  zeros(n1,sysize);
            zeros(sysize,m1-2*sysize)   alpha(i,1)*ident3   alpha(i,2)*ident3 alpha(i
      end
         [sizegn,sizegm]=size(myG1);
           [sizeA1n,sizeA1m]=size(myA1);
              [n1,m1]=size(myA);
              [Gn,Gm]=size(myG2)
              zerosG=zeros( Gn,Gm)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Create the bottom row of the A matrix.
%  This will need changing per problem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 myA1=[myA;
      myG1  zeros(sizeA1n, m1-sizeA1m-sizegm)  myA1];
  myA2=[myA;
       myG2   zeros(sizeA1n,m1-sizeA1m-sizegm)  myA2 ];

myB= zeros(1,(rho )*sysize );
myB1=[myB'
   myB1];
myB2=[myB'
   myB2];
           [sizem1n,sizem1m]=size(myM1);
myM=zeros((rho )*sizem1n,sizem1m );
myM1=[myM;
   myM1];
myM2=[myM;
   myM2];
           [sizec1n,sizec1m]=size(myC1);

myC=zeros(sizec1n,(rho )*sizec1m);
myC1=[myC myC1];
myC2=[myC myC2];



A0= myA1
A1= myA2
B0=myB1
```

```
B1=myB2
C0=myC1
C1=myC2
M0=myM1
M1=myM2
N0=myN1
N1=myN2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  This is the change in coordinates taken for Kirk's thesis.
%may need to do constant orthogonal change of coords on the noise
%do it via a QR decomp on N_i^T to get [Nb_i 0],
%where Nb_i is invertible, also gives [Mb_i Mt_i]
[Q0,R0]=qr(N0')
[Q1,R1]=qr(N1')
pause
%now N_i^T = Q_i * R_i, so N_i = R_i^T * Q_i^T
%and Q_i^T is an orthogonal matrix
%absorb Q_i^T into the noise vector nu_i to get new noise vector
%and Nb_i becomes the invertible part of R_i^T
%may need to fix signs in Q_i and/or R_i
R0=-R0;
R1=-R1;
Q0(1,1)=-Q0(1,1);
Q0(2,2)=-Q0(2,2);
Q1(1,1)=-Q1(1,1);
Q1(2,2)=-Q1(2,2);
%break down into Mb_i, Mt_i, Nb_i, and Nt_i (zeros)
[mN0,nN0]=size(N0);
[mN1,nN1]=size(N1);
mnN0=min(mN0,nN0);
```

```
mnN1=min(mN1,nN1);
mxN0=max(mN0,nN0);
mxN1=max(mN1,nN1);
N0n=R0';
N1n=R1';
Nb0=N0n(1:mnN0,1:mnN0);
Nb1=N1n(1:mnN1,1:mnN1);
Nt0=N0n(:,mnN0+1:mxN0); %just need the size of this
Nt1=N1n(:,mnN1+1:mxN1); %and this
M0n=M0*Q0';
M1n=M1*Q1';
Mb0=M0n(:,1:mnN0);
Mb1=M1n(:,1:mnN1);
Mt0=M0n(:,mnN0+1:mxN0);
Mt1=M1n(:,mnN1+1:mxN1);
%create reduced model system matrices
[mA0,nA0]=size(A0);
[mA1,nA1]=size(A1);
[mMt0,nMt0]=size(Mt0);
[mMt1,nMt1]=size(Mt1);
[mNb,nNb]=size(Nb1);
[mNt0,nNt0]=size(Nt0);
[mNt1,nNt1]=size(Nt1);
A=[A0-Mb0*inv(Nb0)*C0 Mb0*inv(Nb0)*C1; zeros(mA1,nA0) A1]
M=[Mt0 Mb0*inv(Nb0)*Nb1 zeros(mMt0,nMt1); zeros(mMt1,nMt0) Mb1 Mt1]
B=[B0;B1]
C=[inv(Nb0)*C0 -inv(Nb0)*C1];
N=inv(Nb0)*[zeros(mNb,nNt0) Nb1 zeros(mNb,nNt1)];
%Q and H without beta
Qnb=2*C'*C
```

```
Hnb=-4*C'*N
%size of upper left block of R
ulident=nNt0
%Nb1'Nb0'^-1Nb0^-1Nb1 for the center block of R
Nb1
Nmess=Nb1'*inv(Nb0')*inv(Nb0)*Nb1
[nnmess,mnmess]=size(Nmess)
%size of lower right block of R
lrident=nNt1
rtop=[eye(ulident,ulident) zeros(lrident, nnmess+lrident)]
rmiddle= [zeros(nnmess,ulident) Nmess zeros(nnmess,lrident)]
rbottom=[zeros(ulident, nnmess+ulident) eye(lrident,lrident)]
disp('the constants in Rtogether are for easy search and replacing')
rtogether=[45*rtop;78*rmiddle;99*rbottom]

%end of routine
if(1)
save q.txt Qnb -ascii

save b.txt B -ascii

save c.txt C -ascii

save m.txt M -ascii

save r.txt rtogether -ascii

save n.txt Nmess -ascii

save a.txt A -ascii
save h.txt Hnb -ascii
end
```

A.4.2 Differences Driver

The following program solves the first example using the difference approximation.

```
       PROGRAM prog3

      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)

      INTEGER NIW,MAXPHS,NW,MAXCS,MAXDP
      PARAMETER (NIW =20050000,MAXPHS = 5,NW = 20005000,MAXCS=1000500)
      PARAMETER (MAXDP = 1000)
      INTEGER IW(NIW),IPCPH(MAXPHS+1),IPDPH(MAXPHS+1),NEEDED,IER
       DOUBLE PRECISION W(NW),CSTAT(MAXCS),DPARM(MAXDP)



      EXTERNAL setpoint1IN,dumyig,setpoint1DE,GETTINGvPF,DUMYPR
       call hhsocs('default')
        call hhsnlp('sparse default')
c        write(*,*) 'running'
         open (55, FILE='prog3.dat', STATUS='OLD')
         CALL HHSOCS('ipuocp=55')
c      CALL HHSOCS('socout=i9j9k9')
      CALL HHSOCS('IPODE=20')
      CALL HHSOCS('IPNLP=20')
      call hhsocs('ipgrd=20')
c       CALL HHSOCS('socout=i9a9b9c9d9e9f9g9h9j9k9l9m9n9o9p9q9r9')
      CALL HHSNLP('MAXNFE=5000000')
      CALL HHSOCS('mxpcon=100')
      CALL HHSNLP('NITMAX=500')
c      CALL HHSNLP('KTOPTN=SMALL') c      CALL HHSNLP('tolpvt=.1')
      CALL HHSOCS('MXSTAT=300')
      CALL HHSNLP('IOFLAG=20')
c     CALL HHSNLP('ALGOPT=FM')
      CALL HHSOCS('MITODE=20')
      CALL HHSOCS('ODETOL=1.D-7')
      CALL HHSOCS('SPRTHS=SPARSE')
      CALL HHSOCS('MTSWCH=3')
      CALL HHSOCS('NSSWCH=1')
C      CALL HHSOCS('ITSWCH=2') c      CALL Hhsnlp('CONTOL=1.D-6')
c     CALL HHSNLP('OdeTOL=1.D-4')
      call hhsocs('mxterm=200')
c-------define parameters---------------- c we have the problem
x'=ax(i)+bu(i)+gx(i-1)..kinda...see prob notes
      CALL HDSOCS(setpoint1IN,dumyig,setpoint1DE,GETTINGvPF,DUMYPR,
     +            IW,NIW,W,NW,MAXPHS,CSTAT,MAXCS,IPCPH,DPARM,MAXDP,
     +            IPDPH,NEEDED,IER)
      write(*,*) '****************************************'
      write(*,*) 'main prgram2'
      write(*,*) 'error code', ier
      write(*,*) 'needed', needed
      write(*,*) '****************************************'


      close(55)



       END
C************************************************************
c************************************************************
```

```
      SUBROUTINE setpoint1IN(IPHASE,NPHS,METHOD,NSTG,NCF,NPF,NPV,NAV,
     +                  NGRID,INIT,MAXMIN,MXPARM,P0,PLB,PUB,PLBL,
     +                  MXSTAT,Y0,Y1,YLB,YUB,STSKL,STLBL,MXPCON,CLB,
     +                  CUB,CLBL,MXTERM,COEF,ITERM,TITLE,IER)

      implicit double precision (a-h,o-z)



      INTEGER IPHASE,NPHS,METHOD,NSTG,NCF(3),NPF(2),NPV,NAV,NGRID,
     +        INIT,MAXMIN,MXPARM,MXSTAT,MXPCON,MXTERM,ITERM(4,MXTERM),
     +        IER,NTERM,NPATH

      DOUBLE PRECISION P0(MXPARM),PLB(MXPARM),PUB(MXPARM),Y0(0:MXSTAT),
     +          Y1(0:MXSTAT),YLB(-1:1,0:MXSTAT),YUB(-1:1,0:MXSTAT),
     +          STSKL(0:MXSTAT+MXPARM,2),CLB(MXPCON),CUB(MXPCON),
     +          COEF(MXTERM)

      CHARACTER PLBL(MXPARM+2)*80,STLBL(0:MXSTAT)*80,
     +          CLBL(0:MXPCON)*80,TITLE(3)*60

       integer z1,z2,mu1 ,lam1,lam2,bigz,v

      write(*,*) '****************************************'
      write(*,*) 'initialization'
      write(*,*) '****************************************'

       z1=0
       z2=6
       lam1=12
       lam2=18
       bigz=24
       mu1=25
       v=28
       write(*,*) 'z1=', z1,' z2=', z2
       write(*,*) 'lam1=', lam1,' lam2=', lam2
       write(*,*) 'bigz=', bigz,' mu1=', mu1
       write(*,*) ' v=', v



C  Number of differential equations
      NCF(1) =  25

c  Number of algebraic equations
      NCF(2)=3

c Number of quadrature equations
      NCF(3)=1

c  Number of algebraic variables
      NAV = 4

c Optimization flag -1..min  0..fease...1..max
      maxmin=-1

c  Number of parameters
      npv=1

      write(*,*) ' numstateeq=', ncf(1), ' TFinal= ', TFinal
      write(*,*) 'num algeq=', ncf(2),'numcont=', nav

c Initial Grid size
```

```
      NGRID=13

c type of initial guess
      INIT = 1
c number of point functions
      npf(1)=13
      npf(2)=0
C     ----GUESS FOR INITIAL TIME AND BOUNDARY CONDITION c
---------the beginning and final times are fixed. c

      Y0(0) = 0
      Y1(0) = 2
      YLB(-1,0) = Y0(0)
      YUB(-1,0) = Y0(0)
      YLB(1,0) = Y1(0)
      YUB(1,0) = Y1(0)
      write(*,*) 'y1', y1(0)

C----DEFINE INITIAL guess and bounds CONDITIONS FOR STATE
VARIABLES c---on the first and last interval, we want psi minus
and psi plus c----------set everything to 0 c
        DO 150 I=1, ncf(1)+nav
        Y0(I) =10
        Y1(I) =50
 150    CONTINUE

c--------lamdas c-------set lam(omega)=0
        DO 152 I=1, 6
        YLB( 1,lam1+i)= 0
        YUB( 1,lam1+i)= 0
c
```

```
        YLB( 1,lam2+i)= 0
        YUB( 1,lam2+i)= 0



 152    CONTINUE
c-------Z(h)>=1
        YLB(1,bigz+1)= 1

c---------------set the parameter

          p0(1)=.5

c-------.01<=beta<=.99
       PLB(1)= .41
       PUB(1)= .69
       plbL(1)='BETA    BETA'

c*******************************************************
c*******************************************************
c******************************************************* C-------
Describes inequality  or algebraic constraint c-------  these
constraints are described in F

C  defined an a loop

        NTERM=0
        NPATH=0
c--------define the terms that are the path constraints
      DO 800 I=1,ncf(2)
         NPATH = npath+1
         nterm=nterm+1
c----calculate second point function c------------define what
quantity term j belongs to

         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in

         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1

         ITERM(3,NTERM) = 0

c--------------define which constraint this will be used in
         ITERM(4,NTERM) = -i
         CLB(i)=0
         cub(i)=0
         COEF(NTERM) = 1.


  800    CONTINUE

c*******************************************************
C*****************************************************
        DO 801 I=2,npf(1)
      NPATH = npath+1

         nterm=nterm+1
c----lambda1(0)-.5beta*x1(0)=0 c---this is evaluated in a point
```

```
function c------------define what quantity term j belongs to

         ITERM(1,NTERM) = npath

c-------------define which phase term j is computed in

         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1

         ITERM(3,NTERM) = -1
c------define which function this is c------since
         ITERM(4,NTERM) =  -i
         CLB(npath)=0
         cub(npath)=0

         COEF(NTERM) = 1.
  801   continue
C******************************************************
c---------calculate Z(0)-blah=0

       NPATH = npath+1
         nterm=nterm+1
c----calculate first point function c------add point function to
the cost

c-------first term is for the x(0) term c------------define what
quantity term j belongs to

         ITERM(1,NTERM) = npath
```

```
c-------------define which phase term j is computed in

         ITERM(2,NTERM) = 1

c-------------define where in the phase the term is computed
c-------------  t0=-1, during=0, tfinal=1

         ITERM(3,NTERM) = -1
c-------------
         ITERM(4,NTERM) =  -1
         CLB(NPATH)= 0
          cub(NPATH)=0
C*****************************************************
C***************************************************** C
Describes objective quadrature function
      NCF(3) = 1
      NTERM = NTERM + 1
      NPATH = NPATH+1
       ITERM(1,NTERM) = 0
       ITERM(2,NTERM) = 1
       ITERM(3,NTERM) = 0
       ITERM(4,NTERM) = -(ncf(2)+1)
       COEF(NTERM) = 1.


      RETURN
      END
```

```
SUBROUTINE setpoint1DE(IPHASE,T,Z,NZ,P,NP,F,NF,IFERR)

implicit double precision (a-h,o-z)

INTEGER IPHASE,NZ,NP,NF,IFERR

DOUBLE PRECISION T,Z(NZ),P(NP),F(NF)
DOUBLE PRECISION t3,t8,t19,t24,t34,t38
DOUBLE PRECISION t41,t43,t45,t46,t47,t48
DOUBLE PRECISION t50,t54,t58,t61,t63, t67
DOUBLE PRECISION t74,t76,t77,t78,t88,t89
t3 = 0.25E1*Z(3)
 t8 = 0.25E1*Z(4)
t19 = 0.25E1*Z(9)
t24 = 0.25E1*Z(10)
t34 = 0.25E1*Z(14)
t38 = 0.25E1*Z(15)
t41 = 0.25E1*Z(16)
t43 = 0.25E1*Z(17)
t45 = P(1)*Z(6)
t46 = 2*t45
t47 = P(1)*Z(12)
t48 = 2*t47
t50 = 2*P(1)*Z(27)
t54 = 0.25E1*Z(20)
t58 = 0.25E1*Z(21)
t61 = 0.25E1*Z(22)
t63 = 0.25E1*Z(23)
t67 = t45-t47
t74 = Z(26)**2
t76 = Z(27)**2
```

```
t77 = Z(28)**2
t78 = 1-P(1)
t88 = abs(Z(29))
t89 = t88**2
F(1) = -0.75E1*Z(1)+10*Z(2)-t3
F(2) = -0.25E1*Z(1)+t3
F(3) = -0.25E1*Z(2)+t8
F(4) = -t3+0.25E1*Z(5)
F(5) = -t8+0.25E1*Z(6)
F(6) = 0.1E1*Z(1)-0.3E1*Z(6)+Z(29)-Z(26)
F(7) = -0.75E1*Z(7)+10*Z(8)-t19
F(8) = -0.25E1*Z(7)+t19
F(9) = -0.25E1*Z(8)+t24
F(10) = -t19+0.25E1*Z(11)
F(11) = -t24+0.25E1*Z(12)
F(12) = 0.1E1*Z(7)-0.2E1*Z(12)+Z(29)-Z(28)
F(13) = 0.75E1*Z(13)+t34-Z(18)
F(14) = -10*Z(13)+t38
F(15) = 0.25E1*Z(13)-t34+t41
F(16) = -t38+t43
F(17) = -t41
F(18) = -t46+t48+t50-t43+0.3E1*Z(18)
F(19) = 0.75E1*Z(19)+t54-Z(24)
F(20) = -10*Z(19)+t58
F(21) = 0.25E1*Z(19)-t54+t61
F(22) = -t58+t63
F(23) = -t61
F(24) = t46-t48-t50-t63+0.2E1*Z(24)
F(25) = t67*Z(6)-t67*Z(12)-2*t67*Z(27)+t74*P(1)+t76+t77*t78
F(26) = 2*Z(26)*P(1)-Z(18)
F(27) = 2*Z(27)-2*t45+2*t47
```

```
      F(28) = 2*Z(28)*t78-Z(24)
      F(29) = t89
      RETURN
      END




       SUBROUTINE GETTINGvPF(IPHASE,IPHEND,T,Z,NZ,P,NP,PSI,NPSI,IFERR)

       INTEGER  IPHASE,IPHEND,NZ,NP,NPSI,IFERR
       double precision T,Z(NZ),P(NP),PSI(NPSI)
       double precision t1,t4,t7 ,t10,t13,t16
       double precision t21,t23,t26,t29,t32,Q1,Q2
       double precision t32,t35,t38,t19,t39

       double precision lamconst
       double precision gamma13,gamma14,gamma15,gamma16
       double precision gamma18,gamma19,gamma20,gamma11
       double precision gamma12,gamma17,Q1,Q2

       Q1=1
       Q2=1
    lamconst=1
    gamma11=.2
    gamma12=.2
    gamma13=.2
    gamma14=.2
    gamma15=.2
    gamma16=.2
    gamma17=.2
    gamma18=.2
    gamma19=.2
    gamma20=.2


c-------my c---beta x_0(0)^T P x_0(0) + (1-beta) x_1(0)^T P x_1(0)
      if (iphend.eq.-1) then
      t1 = Z(1)**2
      t4 = Z(2)**2
      t7 = Z(3)**2
      t10 = Z(4)**2
      t13 = Z(5)**2
      t16 = Z(6)**2
      t19 = Z(7)**2
      t21 = 1-P(1)
      t23 = Z(8)**2
      t26 = Z(9)**2
      t29 = Z(10)**2
      t32 = Z(11)**2
      t35 = Z(12)**2
      t38 = t1*gamma11*P(1)+t4*gamma12*P(1)+t7*gamma13*P(1)+t10*gamma14*
     #P(1)+t13*gamma15*P(1)+t16*Q1*P(1)+t19*gamma16*t21+t23*gamma11*t21+
     #t26*gamma18*t21+t29*gamma19*t21+t32*gamma20*t21+t35*Q2*t21
      t39 = 2*t38
      t32 = 1-P(1)
      psi(1) = t39-Z(25)
      psi(2) = Z(14)+2*lamconst*Z(2)*gamma12*P(1)
      psi(3) = Z(15)+2*lamconst*Z(3)*gamma13*P(1)
      psi(4) = Z(16)+2*lamconst*Z(4)*gamma14*P(1)
      psi(5) = Z(17)+2*lamconst*Z(5)*gamma15*P(1)
      psi(6) = Z(18)+2*lamconst*Z(6)*Q1*P(1)
      psi(7) = Z(19)+2*lamconst*Z(7)*gamma16*t32
      psi(8) = Z(20)+2*lamconst*Z(8)*gamma11*t32
      psi(9) = Z(21)+2*lamconst*Z(9)*gamma18*t32
      psi(10) = Z(22)+2*lamconst*Z(10)*gamma19*t32
      psi(11) = Z(23)+2*lamconst*Z(11)*gamma20*t32
      psi(12) = Z(24)+2*lamconst*Z(12)*Q2*t32
      psi(13) = Z(13)+2*lamconst*Z(1)*gamma11*P(1)

      endif

      end
```

List of References

[1] Panos J. Antsaklis and Anthony N. Michel. *Linear Systems.* McGraw-Hill, 1997.

[2] Michael Athans and Peter L. Falb. *Optimal Control: An Introduction to the Theory and Its Applications.* McGraw-Hill Book Company, 1966.

[3] H. T. Banks. Approximation of delay systems with applications to control and identification. In *Functional differential equations and approximation of fixed points (Proc. Summer School and Conf., Univ. Bonn, Bonn, 1978)*, volume 730 of *Lecture Notes in Math.*, pages 65–76. Springer, Berlin, 1979.

[4] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application.* Information and System Science Series. Prentice-Hall, 1993.

[5] J. T. Betts. *Practical Methods for Nonlinear Control Using Nonlinear Programming.* Number 3 in Advances in Design and Control. SIAM, 2000.

[6] J. T. Betts and P. D. Frank. A sparse nonlinear optimization algorithm. *Journal of Optimization Theory and Applications*, 82:519–541, 1994.

[7] J. T. Betts and W. P. Huffman. Sparse Optimal Control Software socs. Mathematics and Engineering Analysis Technical Document mea-lr-085-r1, Boeing Information and Support Services, July 2000.

[8] S. Campbell, K. Drake, and R. Nikoukhah. Auxillary system design for multimodel identification in systems with multiple delays. In *Proc. 10th Mediterranean Conference on Control and Automation*, Lisbon, July 7-11 2002.

[9] S. Campbell, K. Drake, and R. Nikoukhah. Early decision making when using proper auxillary signals. In *Proc. IEEE Conference on Decision and Control*, pages 1832–1837, Las Vegas, Nevada, Dec.10-13 2002.

[10] S. Campbell, K. Drake, and R. Nikoukhah. Analysis of spline based auxiliary signal design for failure detection in delay systems. In *Proc. IEEE CSMC*, Washington, DC, October 2003.

[11] S. Campbell, K. Drake, R. Nikoukhah, and F. Delebecque. Rapid multi-model identification in systems with delays. In *3rd IFAC Workshop on Time Delay Systems (TDS 2001)*, pages 296–301, Sante Fe, New Mexico, Dec. 8-10 2001.

[12] S. Campbell and R. Nikoukhah. *Auxiliary Signal Design for Failure Detction.* Princeton University Press, 2004, In Press.

[13] S. L. Campbell, K. E. Brenan, and L Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* SIAM, 1996.

[14] S. L. Campbell, K. Horton, R. Nikoukhah, and F. Delebecque. Rapid model selection and the separability index. *Proc. SAFEPROCESS*, 2000.

[15] S. L. Campbell, K. Horton, R. Nikoukhah, and F. Delebecque. Optimaization formulations of auxiliary signal desing for rapid multi-model identification. *Automatica*, 38:1313–1325, 2002.

[16] S. L. Campbell and R. Nikoukhah. Auxiliary design for robust active failure detection: the general cost case. *Proc. SAFEPROCESS*, 2003.

[17] Rong Chen and Kenneth A. Loparo. Identification of time delays in linear stochastic systems. *Internat. J. Control*, 57(6):1273–1291, 1993.

[18] Moody Chu. *Notes for MA780.* World Wide Web, http://www4.ncsu.edu/∼ mtchu/ Teaching/ Lectures/ MA530/ ma780.html, 1996–2003.

[19] Anastasios Delopoulos and Georgios B. Giannakis. Consistent identification of stochastic linear systems with noisy input-output data. *Automatica J. IFAC*, 30(8):1271–1294, 1994.

[20] J. E. Dennis and Robert B. Schanabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Number 16 in Classics in Applied Mathematics. SIAM, Philadelphia, PA, 1996.

[21] Paul M. Frank. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy -a survey and some new results. *Automatica*, 26(3):459–474, May 1990.

[22] H. Górecki, S. Fuksa, P. Grabowski, and A. Korytowski. *Analysis and Synthesis of Time Delay Systems.* John Wiley & Sons, 1989.

[23] J. J. Gertler. Survey of model-based failure detection and isolation in complex plants. *IEEE Control Systems Magazine*, 1988.

[24] C. Glader, G. Högnäs, P. M. Mäkilä, and H. T. Toivonen. Approximation of delay systems -a case study. *Internat. J. Control*, 53(2):369–390, 1991.

[25] H.J. Greenberg. *Mathematical Programming Glossary.* World Wide Web, http://www.cudenver.edu/∼hgreenbe/glossary/, 1996–2003.

[26] Xing Sheng Gu and Yang Zeng Hu. System analysis and parameter identification of linear time-varying delay systems via piecewise multiple Chebyshev polynomials. *Control Theory Appl.*, 8(2):154–162, 1991.

[27] Jack Hale and S.M.V Lunel. *Introduction to Functional Differential Equations*, volume 99. Springer Verlag, 1993.

[28] A. M. Hardie. *The Elements of Feedback and Control.* Oxford University Press, 1964.

[29] Ferenc Hartung and Janos Turi. Identification of parameters in delay equations with state-dependent delays. *Nonlinear Anal.*, 29(11):1303–1318, 1997.

[30] Kirk Horton. *Fault Detection And Model Identification In Linear Dynamical Systems.* PhD thesis, North Carolina State University, 2001.

[31] Waterloo Maple Inc. http://www.mathworks.com/.

[32] R. Isermann. Process fault detection based on modeling and estimation methodsa survey. *Automatica*, 20(4):387404, July 1984.

[33] K. Ito and F. Kappel. Approximation of infinite delay and Volterra type equations. *Numer. Math.*, 54(4):405–444, 1989.

[34] K. Ito and F. Kappel. A uniformly differentiable approximation scheme for delay systems using splines. *Applied Mathematics and Optimization*, 23:217–262, 1991.

[35] Kazufumi Ito and Franz Kappel. Approximation of semilinear Cauchy problems. *Nonlinear Anal.*, 24(1):51–80, 1995.

[36] Kazufumi Ito and Franz Kappel. *Evolution equations and approximations*, volume 61 of *Series on advances in mathematics for applied sciences.* World Scientific Pub, River Edge, NJ, 2002.

[37] Kazufumi Ito and Franz Kappel. *Evolution equations and approximations*, volume 61 of *Series on Advances in Mathematics for Applied Sciences.* World Scientific Publishing Co. Inc., River Edge, NJ, 2002.

[38] F. Kappel. Spline approximation for autonomous nonlinear functional-differential equations. *Nonlinear Anal.*, 10(5):503–513, 1986.

[39] F. Kappel. Approximation of LQR-problems for delay systems: a survey. In *Computation and control, II (Bozeman, MT, 1990)*, volume 11 of *Progr. Systems Control Theory*, pages 187–224. Birkhäuser Boston, Boston, MA, 1991.

[40] C. T. Kelley. *Iterative Methods For Linear and Nonlinear Equations.* Frontiers In Applied Mathematics. SIAM, 1995.

[41] Erwin Kreyszig. *Introductory Functional Analysis with Applications.* John Wiley and Sons, 1989.

[42] T. T. Lee and S. C. Tsay. Approximate solutions for linear time-delay systems via the Padé approximation and orthogonal polynomials expansions. *Control Theory Adv. Tech.*, 3(2):111–128, 1987.

[43] Cornelius T. Leondes and Edward C. Wong. An identification algorithm for linear stochastic systems with time delays. *Internat. J. Control*, 36(3):445–459, 1982.

[44] Frank L. Lewis and Vassilis L. Syrmos. *Optimal Control*. John Wiley & Sons, New York, 2nd edition, 1995.

[45] David G. Luenberger. *Otimization by Vector Space Methods*. John Wiley & Sons, 1969.

[46] P. M. Mäkilä and J. R. Partington. Laguerre and Kautz shift approximations of delay systems. *Internat. J. Control*, 72(10):932–946, 1999.

[47] P. M. Mäkilä and J. R. Partington. Shift operator induced approximations of delay systems. *SIAM J. Control Optim.*, 37(6):1897–1912 (electronic), 1999.

[48] A. Manitius and H. Tran. Numerical simulation of a non-linear feedback controller for a wind tunnel model involving a time delay. *Optimal Control Applications and Methods*, 7:19–39, 1986.

[49] The MathWorks. http://www.maplesoft.com/.

[50] R. Nikoukhah, S. L. Campbell, and F. Delebecque. Detection signal design for failure detection: a robust approach. *Int. J. Adaptive Control and Signal Processing*, 14:701–724, 2000.

[51] R. Nikoukhah, S. L. Campbell, K. Horton, and F. Delebecque. Auxiliary signal design for robust multi-model identification. *IEEE Tran. Aut. Cont*, 47:158–163, 2002.

[52] R. Nikoukhah, F. Delebecque, S. L. Campbell, and K. Horton. Multi-model identification and the separability index. *Proc. Mathematical Theory Networks & Systems*, 2000.

[53] Ramine Nikoukhah. Innovations generation in the presence of unknown inputs: application to robust failure detection. *Automatica J. IFAC*, 30(12):1851–1867, 1994.

[54] Jonathan R. Partington. Approximation of delay systems by Fourier-Laguerre series. *Automatica J. IFAC*, 27(3):569–572, 1991.

[55] Linda R. Petzold and Uri M. Ascher. *Computer Methods for Ordinary Differential-Algebraic Equations.* SIAM, 1998.

[56] Farzad Pourboghrat and Dong Hak Chyung. Parameter identification of linear delay systems. *Internat. J. Control*, 49(2):595–627, 1989.

[57] W. Prager. A one parameter family of spline-type schemes for approximation of delay systems. *J. Math. Anal. Appl.*, 177(1):135–165, 1993.

[58] A. V. Savkin and I. R. Petersen. New approach to model validation and fault diagnosis. *J. Optim. Theory Appl.*, 94(1):241–250, 1997.

[59] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Number 12 in Texts in Applied Mathematics. Springer-Verlag, second edition, 1991.

[60] Lloyd N. Trefethen and Martin H. Gutknecht. The Carathéodory-Fejér method for real rational approximation. *SIAM J. Numer. Anal.*, 20(2):420–436, 1983.

[61] Elvio Vidal Castedo, José R. Perán González, and Rafael Grossi Calleja. Recursive estimation and delay processes subject to important stochastic perturbations. *Rev. Inform. Automát.*, 21(3):22–30, 1988.

[62] P. Zhang, S. X. Ding, G. Z. Wang, and D. H. Zhou. Fault detection for multirate sampled-data systems with time delays. *Internat. J. Control*, 75(18):1457–1471, 2002.

[63] Wei Xin Zheng and Chun Bo Feng. An optimizing search-based identification method for stochastic time-delay systems. *Inform. and Control (Shenyang)*, 18(1):19–26, 1989.

[64] Wei Xing Zheng and Chun Bo Feng. Identification of stochastic time lag systems in the presence of colored noise. *Automatica J. IFAC*, 26(4):769–779, 1990.