ABSTRACT

FAIR, MARTENE L. Active Incipient Fault Detection With Multiple Simultaneous Faults. (Under the direction of Dr. Stephen Campbell).

Recently an active approach for the detection of incipient faults has been introduced. This approach is based on a multi-model formulation of normal and faulty systems, where the fault is modeled as a change in a system parameter. It involved finding an auxiliary signal designed to enhance the detection of variations in this parameter. However, that work examined systems where there was only one incipient fault occurring at a time.

We detect small parameter variations in linear uncertain systems due to incipient faults using an active approach to fault detection. Unlike previous studies where it is usually assumed that there is only one fault, we allow for multiple faults to occur simultaneously which is a natural assumption in the incipient case. This dissertation is an extension of the multi-model approach used for the construction of auxiliary signals for incipient failure detection.

We examine the discrete time case with a modified noise bound, which is a problem that can be attacked directly. We also study the continuous time case which has a similar model, but requires that we solve a linear quadratic regulator problem. Both models involve only additive uncertainty. However, in our setup of the incipient fault problem we find that some types of model uncertainty are allowable. In essence, some parameters are treated as faults and some of them are treated as model uncertainties.

Active Incipient Fault Detection With Multiple Simultaneous Faults

by
Martene L. Fair

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fullfillment of the
requirements for the Degree of
Doctor of Philosophy

Mathematics

Raleigh, North Carolina

2010

APPROVED BY:

Dr. Ernest Stitzinger	Dr. Robert White
Dr. Stephen Campbell Chair of Advisory Committee	Dr. Negash Medhin

DEDICATION

For my beloved parents, Martha and Adell Fair.

BIOGRAPHY

Martene Fair is a native of Milwaukee, Wisconsin. She attended Milwaukee High School of the Arts (MHSA) and majored in instrumental jazz piano. In May 2001 she graduated from MHSA with honors.

She continued her education at Tennessee State University (TSU) in Nashville, Tennessee. As an undergraduate she was involved in several extracurricular activities. She played college tennis throughout her tenure at TSU and also was an active member in several other organizations. In addition, she served as a mentor for the Packard Science Summer Institute in 2003 and during the summer of 2004 she interned at the National Science Foundation. She earned a Bachelor of Science Degree in Mathematics with certification to teach at the secondary level from Tennessee State University in May of 2005.

She began her graduate studies in mathematics at North Carolina State University (NCSU) in the fall of 2005 as a National Physical Science Consortium Fellow and this afforded her two internship experiences at the National Security Agency. In May of 2008 she received a Master of Science Degree in Mathematics from North Carolina State University. Upon completion of the mathematics doctoral degree, she will begin a tenure-track faculty position at the university level.

ACKNOWLEDGMENTS

First and foremost, I would like to thank God for blessing me with so many great opportunities throughout my life. He is my source and without Him all of my previous accomplishments as well as achieving this goal would not have been possible.

I express my gratitude to my advisor and mentor, Dr. Stephen Campbell. Without his expertise, guidance, and encouragement I would not have been able to complete this research project.

I am also thankful to my research committee: Dr. Negash Medhin, Dr. Ernest Stitzinger, Dr. Robert White, and Dr. Donald Bitzer. I appreciate the support and guidance each one of you provided throughout this process. It has been an awesome learning experience inside and outside of the classroom and I have grown both academically and professionally during my tenure at NCSU.

I appreciate the advice, support, and encouragement I have received from mentors and professors along the way. A special thank you to Dr. Jeanetta Jackson and Dr. Jessie DeAro.

I would also like to thank all of my family and friends for their support, encouragement, and prayers over the years. You have truly made a positive difference in my life.

Finally, thank you mom and dad for your love, support, and belief in me. The positivity and encouragement you both have shown throughout my life are priceless. Thanks for being amazing parents and for giving so much to me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

We live in a society where people are directly and indirectly influenced by sophisticated and highly-automated systems and the smooth operation of these systems is tied to their efficiency and productivity. Therefore, it is important to be knowledgeable of any practical system and its operation.

A fault is an undesirable state of a system, which disrupts the system while it performs its required and expected functions. Therefore, in any practical system it is important to be able to detect faults. The main goal of fault detection is early recognition of problem-prone behavior in an observed system in order to prevent the system from shutting down or causing a problematic incident. Fault detection increases system safety, reliability, and availability.

Fault detection has been the subject of many studies because it plays a pivotal role in various system operations. It is an integral part of the operation of many different systems such as electrical components, mechanical devices, and chemical reaction processes ranging from household utilities to industrial processes. Failure detection has been used to improve the navigation of systems such as planes, boats, and other moving objects as well as in the development of biomedical systems.

The goal of fault detection is the early detection of potential problems that may occur in a system that may cause a breakdown of its functions or other undesired or unexpected results. One issue that we must address when performing fault detection methods is the fact that imperfections and uncertainty in the model as well as noise in the model must be taken into account when analyzing the system for errors and problems.

Due to its importance there has been considerable prior work done on fault detection [23, 36, 43]. In our work we look at the case where multiple faults occur simultaneously. In this study, we first examine the outputs of two difference equation models of a particular system. One model represents the normal behavior of the system and the other model represents its faulty behavior. We want to determine if anything has gone wrong as the system goes through its processes.

We also study the continuous time case and we examine the outputs of two different ordinary differential equation (ode) models of a particular system. Again, we look at two models: one model represents the normal behavior of a system and the other model represents its faulty behavior. We want to determine if a fault has occurred as the system goes through its processes.

There are two main approaches to fault detection. In a passive approach the detector measures system states and looks for a deviation from a known fault-free condition. Using this method the state of the system can be tested by creating an observer or performing statistical analysis on the output of the system. Passive detection is particularly useful for systems that are restrictive to auxiliary signals due to safety, material, or sensitivity issues. One disadvantage to the passive approach is the fact that certain faults can be masked as the system goes through its normal operations. For example, if a particular part of a system is experiencing wear but that part is not in use, this fault may be undetected using the passive method. Some of the work done on the passive approach can be found in [23].

The active approach requires that we do something to the outputs in order to detect faults. Using this method we apply a test signal v to the system to detect the fault, but we also do as little as possible to affect the system's normal operation. We are not doing continual monitoring, but we perform a specific test designed to reveal the presence of incipient faults that minimally disturbs the system. The system disturbance is represented by a cost function that will allow us to determine the optimal input v required that will satisfy our conditions.

1.2 Background

The type of fault we study is called an incipient (slowly developing) fault. When an incipient fault begins to develop within a given system, like a mechanical system or an electrical component, the system can still function in a reasonable manner. We can think of this type of fault as a changing parameter.

Incipient faults have been studied in [14, 31, 32]. In [14] incipient fault detection is applied to nonlinear systems in order to improve automated maintenance for early detection of worn equipment in engineering systems. The work in [31] uses a multi-model approach to detect single incipient faults by implementing a fault detector on-line in models which contain both additive and multiplicative uncertainty. Similarly, in [32] on-line implementation of the fault detector in the case where the failure is modeled as a small unknown change in a system parameter of the normal system. This particular study analyzes the additive uncertainty case.

We consider two models. Given a normal and faulty model of a particular system, we want to find a minimal detection signal v. For the discrete time case, the following is the normal model for the incipient problem with parameter θ:

$$x_{k+1} = A(\theta)x_k + B(\theta)v_k + M\mu_k \tag{1.1a}$$

$$y_k = C(\theta)x_k + N\mu_k. \tag{1.1b}$$

When the parameter is θ, then we call the model normal and the model is called faulty if the parameter is given by $\theta + \delta\theta$. In the single fault case, θ is a scalar parameter. In the multi-fault case θ is a vector. We take $\delta\theta$ to be some unknown drift. We assume that the test period is short enough so that the value of θ and $\delta\theta$ can be considered constant for each test period. This means θ is varying slowly. In addition, there are several requirements on the test signal during the test period including the system should continue to operate in a reasonable manner, the test period should be short, and the effect of the detection signal on the system should be minimal. That is, given bounds on the uncertainty, a test signal v is computed that provides guaranteed detection within the bounds set, and it is also minimal. Minimality can be in terms of the size of the test signal, the impact the test signal has on system performance, or a combination of the two measures.

There are multiple ways to solve this problem. One possibility would be to try to find a shape for v that works for many $\delta\theta's$ and scale the size of v corresponding to the desired detection threshold. This is the approach we follow. We consider two cases: the discrete time and continuous time. Both cases include additive uncertainty and multiple parameters.

Incipient fault detection consists of early detection of small variations in system behavior. In our approach, the test signal is used to reveal the presence of the incipient fault. In fact, the auxiliary signal is designed to enhance the detection of variations in the parameter, which is usually done by using an identification scheme to estimate the parameter on-line. As long as our uncertainty bounds and model assumptions hold, the test signal guarantees the correct decision. Any conservatism in the setup of the problem occurs in using a slightly larger than minimal test signal. In addition, we do not assume any particular source of the additive uncertainty. The additive uncertainty can be a combination of model error, colored noise, and measurement error.

When considering incipient faults, the usual assumption is that there is only one fault happening at a given time, but this is not always appropriate. We will examine an active approach for incipient fault detection when there is more than one concurrent fault. The consideration of multiple faults is a major contribution of this thesis.

The discrete time case will be considered first and this approach will then be extended to the continuous time case. We will specify a threshold, which is some combination of the uncertainties, and guarantee detection if this threshold is exceeded. This will indicate that we have a proper test signal.

The problem solved is a min max problem. In principle, the min max problem can be solved by software, like Matlab or SOCS [11]. Algorithms to solve the incipient fault problem for higher state dimensions and longer time horizons are extensions of this work.

1.3 Basic Theory

Models of dynamical systems that include a set of linear differential and algebraic equations (DAEs)

$$Ez' + Fz = f(t) \tag{1.2}$$

where E is a singular square matrix are called linear descriptor systems. Many systems in various applications can be described by linear descriptor systems. Electrical circuit design and network modeling problems are examples of systems that can be modeled in this way [22].

A descriptor system is one possible result of a system design problem, which is the development of a system which accomplishes certain objectives while fulfilling specified constraints. To begin a system design problem the engineer is given objectives that describe the desired performance characteristics of a particular system along with a set of constraints by which the system is bound.

The control problem is a particular type of system design problem, in which the goal is to generate certain outputs from the system to keep the state of the system within certain bounds. For example, an engineer may be asked to design an airplane that uses minimal fuel. The essential elements of such an optimal control problem are

- a mathematical model of the system,

- a desired output,
- a set of admissible controls,
- a performance measure also called a cost functional.

For the remainder of this thesis, we restrict our study to linear time-invariant systems. For the linear time-invariant case there are special forms for the state-space description. These forms are obtained by means of similarity transformations and are designed to reveal those features of a system that are related to the properties of controllability and observability [2].

Consider a continuous time model with a system based on the linear time invariant ode

$$x' = Ax + Bv \tag{1.3a}$$
$$y = Cx, \tag{1.3b}$$

where x, y, and v are the state, output, and control vectors, respectively and our time interval is $t \in [t_0, t_f]$. Systems usually allow for noise or additive uncertainty by adding a term to both equations in (1.3)

$$x' = Ax + Bv + M\mu \tag{1.4a}$$
$$y = Cx + N\mu, \tag{1.4b}$$

where μ is the unknown uncertainty input and matrices M and N are the weight matrices for the state and output noise, respectively. The concepts of controllability, observability, and stability are important factors in studying systems like (1.3).

Control theory is the mathematical study of how to manipulate the parameters affecting the behavior of a system to produce a desired or optimal outcome. It is an interdisciplinary branch of engineering and mathematics that deals with the behavior of dynamical systems. The desired output of a system is called the control objective.

When one or more output variables of a system need to follow a certain objective over time, a controller manipulates the inputs of the system to obtain the desired effect on the output of the system.

There are two main types of control: closed-loop and open loop. A closed loop control computes the input based on the output. Then the state is adjusted to reach a desired condition, which keeps the state of the system within certain bounds. An open loop control also known as a non-feedback control computes the input of the system without using the current state. In this case, the system does not observe the output of the processes it is controlling. Consequently, a true open loop system cannot correct any errors or compensate for disturbances in the system. Our test signals are open loop. The effect of feedback on test signal design is discussed in [7, 8].

We now define the concepts of reachability and controllability. A state x_1 is called reachable at time t_1 if there exists an input that transfers the state of the system $x(t)$ from zero at some prior time t_0 to x_1 at time t_1. A linear system is said to be controllable at t_0 if it is possible to find some input function $v(t)$, defined over $t \in [t_0, t_f]$, which will transfer the initial state $x(t_0)$ to the origin at some finite time $t_1 \in [t_0, t_f], t_1 > t_0$. If this is true for all initial times t_0 and all initial states $x(t_0)$, the system is completely controllable. Our test signal v is a control (input) that will allow us the ability to manipulate the state and hence the outputs.

For some numbers n and k where $k \leq n$ the Cayley-Hamilton Theorem shows that $R(C_k) = R(C_n)$, where $C_n = C$ is the controllability matrix of the system and $C_n = [B, AB, \ldots, A^{n-1}B] \in R^{n \times mn}$, where A and B are taken from a system model such as (3.25) and $R(C)$ is the range space of the controllability matrix. For discrete linear time invariant systems, it is not necessary to take more than k steps in the control sequence because if the transfer cannot be accomplished in k steps, it cannot be accomplished at all, but it is possible to accomplish x_1 in fewer than k steps. See Example 1.1 on pg. 218 of [2]. In general, the solution of the control problem, v_k, is not unique, i.e., there are many inputs which can accomplish the transfer from $x(0) = x_0$ to $x(k) = x_1$ each corresponding to a particular state trajectory. In control problems, particular inputs are frequently selected that, in addition to transferring the state, satisfy additional criteria such as, minimization of a cost function.

A linear system is said to be observable at t_0 if $x(t_0)$ can be determined from the output function y for $t \in [t_0, t_f]$ and $t_0 \leq t_f$. If this is true for all t_0 and $x(t_0)$, the system is completely observable. In order to check the observability of a system we examine the observability matrix, $O = [C, CA, \ldots, CA^{n-1}]^T$, which is constructed using the A matrix and C matrix from (1.3). If O has full column rank, i.e., rank $O = n$, then the system is state observable or (A, C) is observable. If (A, C) in (1.3) is not observable, but the unobservable eigenvalues are stable, then (A, C) is detectable. In other words, a system is detectable if and only if all of its unobservable eigenvalues (modes) are stable. Therefore, even though not all system modes are observable, the ones that are not observable but detectable do not require stabilization.

When designing the mathematical model of a system we must account for the stability of the models. Therefore, before introducing a test signal into the model we often want to make sure that at least within some set of parameters the system is stable. In order to examine a system's stability we must consider its equilibria.

Dynamical systems, occurring in nature or manufactured, usually function in some specified mode. The most common modes are operating points that often turn out to be equilibria.

Most of the time we are interested in the asymptotic stability of an equilibrium (operating point). That is, when the state of a given system is displaced (disturbed) from its desired operating value which we call the equilibrium, the expectation is that the state will return to the equilibrium. For example, a vehicle under cruise control, traveling at the desired constant speed of 50 mph (which determines the operating point or equilibrium condition), encounters perturbations due to hill climbing (hill descending), which will result in decreasing (or increasing) speeds. In a well designed cruise control system, it is expected that the automobile will return to its desired operating speed of 50 mph [2].

Another qualitative characterization of dynamical systems is referred to as input-output stability. This is the expectation that bounded system inputs will result in bounded system outputs and small changes in inputs will result in small changes in outputs. This property is important in tracking systems, where the output of the system is expected to follow a desired reference or target trajectory. Many times it is possible to establish a connection between the input-output stability properties and the Lyapunov stability properties of an equilibrium. This link is well understood in linear systems.

There are many qualitative characterizations that are important to systems theory. These characterizations deal with the various types of stability properties of an equilibrium and are referred to as Lyapunov stability [2].

We start by introducing some important definitions related to local properties of an equilibrium. For the following system

$$x' = f(t, x), \tag{1.5}$$

assume that (1.5) has an isolated equilibrium at the origin. That means $f(t, 0) = 0$ for all $t \geq 0$.

Consider the linear autonomous homogeneous system with t representing time

$$x' = Ax, \quad t \geq 0, \tag{1.6}$$

and the linear homogeneous system

$$x' = A(t)x, \quad t \geq t_0 \geq 0, \tag{1.7}$$

where $A(t)$ is assumed to be continuous. Note that if $A(t)$ is nonsingular for all $t \geq 0$, then $x = 0$ is always an equilibrium of (1.6) and (1.7) and $x = 0$ is the only equilibrium of (1.7). Also, the solution of (1.7) for $x(t_0) = x_0$ is of the form

$$\phi(t, t_0, x_0) = \Phi(t, t_0)x_0, \quad t \geq t_0,$$

where Φ denotes the state transition matrix of any $A(t)$ and the solution of (1.6) for $x(t_0) = x_0$ is given by

$$\phi(t, t_0, x_0) = \Phi(t, t_0)x_0.$$

If A is constant then $\phi(t, t_0, x_0)$ can be written as

$$\begin{aligned} \phi(t, t_0, x_0) &= \Phi(t, t_0)x_0 \\ &= \Phi(t - t_0, 0)x_0 \\ &\equiv \Phi(t - t_0)x_0 \\ &= e^{A(t-t_0)}x_0. \end{aligned} \tag{1.8}$$

Definition 1. *For linear systems*

$$x' = A(t)x, \tag{1.9}$$

a state transition matrix is a matrix whose product with the state vector x at an initial time t_0 gives x at a later time $t > t_0$. The state transition matrix can be used to find the general solution of linear dynamical systems. The state transition matrix is denoted by $\Phi(t, t_0)$. Thus $x(t) = \Phi(t, t_0)x(t_0)$.

Definition 2. *For any $t_0 \geq 0$ that $\phi(t, t_0, x_e) = x_e$ for all $t \geq t_0$, i.e. the equilibrium x_e is a unique solution of (1.5) with initial data given by $\phi(t, t_0, x_e) = x_e$.*

Definition 3. *A point $x_e \in \Re^n$ is called an equilibrium point if $x_e' = f(t, x_e) = 0$ for all $t \geq \tilde{t}$. The equilibrium $x = 0$ of (1.5) is said to be stable if for every $\epsilon > 0$ and any $t_0 \in \Re^+$ there exists a $\delta(\epsilon, t_0) > 0$ such that for all $t \geq t_0$*

$$\|\phi(t, t_0, x_0)\| < \epsilon \tag{1.10}$$

whenever

$$\|x_0\| < \delta(\epsilon, t_0). \tag{1.11}$$

The equilibrium is stable if any other state that starts close enough to the equilibrium stays close to the equilibrium over time.

Definition 4. *The equilibrium $x = 0$ of (1.5) is said to be asymptotically stable if it is stable and for every $t_0 \geq 0$ there exists an $\zeta(t_0) > 0$ such that $\lim_{t \to \infty} \|\phi(t, t_0)\| = 0$ whenever $\|x_0\| < \zeta$.*

The set of all $x_0 \in \Re^n$ such that $\phi(t, t_0, x_0) \to 0$ as $t \to \infty$ for some $t_0 \geq 0$ is called the domain of attraction of the equilibrium $x = 0$ of (1.5) at t_0. Also, when the equilibrium of (1.5) is asymptotically stable, then $x = 0$ is said to be attractive at t_0. That is, an equilibrium is asymptotically stable if any other state is close enough to the equilibrium, then the solution actually converges over time to the equilibrium.

Definition 5. *The equilibrium $x = 0$ of (1.5) is unstable if it is not stable and there exists a $t_0 \geq 0$, an $\epsilon > 0$, and a sequence $x_m \to 0$ of initial points and a sequence t_m*

such that $\|\phi(t_0 + t_m, t_0, x_m)\| \geq \epsilon$ *for all* $m, t_m \geq 0$. *Note that it can happen that a system like (1.5) with unstable equilibrium* $x = 0$ *may have only bounded solutions. The equilibrium* $x = 0$ *is unstable if some state trajectories move away from the equilibrium as* t *goes to infinity.*

The following theorem is related to Lyapunov stability of linear systems.

Theorem 1.1 1. *The equilibrium* $x = 0$ *of (1.7) is stable if and only if the solutions of (1.7) are bounded, i.e., if and only if*

$$\sup_t \|\Phi(t, t_0)\| \equiv k(t_0) < \infty,$$

where $\|\Phi(t, t_0)\|$ *denotes the matrix norm induced by the vector norm used in* R^n *and* $k(t_0)$ *denotes a constant that may depend on the choice of* t_0. *See page 453 of [2] for a proof of this theorem.*

Stability helps us deal with systems that may not be controllable or observable. We can determine the stability of a linear time invariant system by finding the eigenvalues of matrix A of the system model. If all of the eigenvalues of A have negative real part, the equilibrium $x = 0$ is asymptotically stable. The equilibrium $x = 0$ is stable if and only if all of the eigenvalues have nonpositive real parts and every eigenvalue with zero real part has an associated Jordan block of order one. If any of the eigenvalues of A have either positive real part or has zero real part that is associated with a Jordan block of order greater than one, then the equilibrium $x = 0$ is unstable.

A system is stabilizable if all the unstable eigenvalues are controllable. A system is detectable if all unstable eigenvalues are observable. The system can be handled effectively provided all uncontrollable and unobservable modes are stable.

The concept of feedback connects control theory and fault detection. In a feedback control system, the control $v(t)$ is modified based on information about the system. Detectors (or sensors) measure either the system state or output and then pass that information to the controller, which adjusts the control based on the input from the sensors. One goal of feedback compensator design is to improve the performance of the system by eigenvalue placement [2]. As stated earlier, the stability of a system depends on the eigenvalues of matrix A. Therefore, by assigning desirable values to eigenvalues, system stability can be improved. For the state feedback case, the control can be written as

$$v(t) = Fu(t) - Kx(t), \tag{1.12}$$

where F is the feed-forward matrix and K is called the feedback gain matrix. Substituting (1.12) into (1.3), we obtain

$$\begin{aligned} x' &= (A - BK)x + BFu && \text{(1.13a)} \\ y &= Cx. && \text{(1.13b)} \end{aligned}$$

Now, the eigenvalues of the $A - BK$ matrix determine the stability of the system. Given the proper construction of the feedback gain matrix K, the eigenvalues are assigned to desired values. For the output feedback case, the relation for the control is

$$v(t) = Fu(t) - Ky(t), \tag{1.14}$$

where F and K are defined as above. If we substitute this relation into (1.3), we obtain

$$x' = (A - BKC)x + BFu. \tag{1.15}$$

Here the eigenvalues of our system, which determine the system's stability, come from matrix $A - BKC$. However, since matrix C is found in (1.15), the output feedback cannot place all of the eigenvalues of the system, that is we can not totally control the pole placement in this case. This limitation occurs when the rank of KC is less than the rank of K.

Using feedback, the basic tool for many fault detection methods can be developed using an observer. For most systems the only information about the system state is through the output vector, which often does not provide all information about the state of a system. In this case, output feedback is the only direct option, but recall that with this method not all the eigenvalues of the system can be placed where desired. To improve the stability of the system in these cases, the most widely used approach is to reconstruct information about the remaining elements of the state vector by creating an observer of the system. Consider the observer

$$\begin{aligned} \hat{x}' &= A\hat{x} + Bv + L(y - C\hat{x}) \quad (1.16) \\ \hat{y} &= C\hat{x}, \quad (1.17) \end{aligned}$$

where $\hat{x}$ is the observer estimate for the state vector. y is the output from the actual system (1.13) and $C\hat{x}$ is the observer output. If we take the difference of the observed system and the original system and assuming $e = \hat{x} - x$ is the observer error, we obtain

$$e' = (A - LC)e. \quad (1.18)$$

L is arbitrary and (A, C) is observable so we can guarantee that the observer error goes to zero by selecting L so that $A - LC$ is stable. Using this setup, state feedback is possible using an observer estimate for the state vector. Therefore, all the system eigenvalues can be placed where desired and the system is completely controllable. Since the complete state vector is reconstructed by the observer, faults which cause the system states to behave unpredictably or undesirably may be detectable by such an observer if the observer estimate is compared to those elements of the system state vector which are accessible. This fault detection can be accomplished without the use of an observer to affect the feedback compensator of the system.

The continuous time case of the incipient fault problem requires that we solve a linear quadratic regulator (LQR) problem in order to find the optimal detection signal. The theory of optimal control is concerned with operating a dynamic system at minimum cost. The case where the system dynamics are described by a set of linear differential equations and the cost is described by a quadratic functional is called the LQ problem [26]. One of the main results in the theory is that the solution is provided by the LQR in the form of a feedback controller.

The settings of a (regulating) controller governing either a machine or process (like an airplane or chemical reactor) are found by using a mathematical algorithm that minimizes a cost function with weighting factors supplied by the engineer. The cost function is often defined as a sum of the deviations of key measurements from their desired values. In effect this algorithm finds those controller settings that minimize the undesired deviations, like deviations from the desired altitude or process temperature. Often the magnitude of the control action itself is included in this sum as to keep the energy expended by the control action itself limited.

Essentially, the LQR algorithm takes care of the tedious work done by the control systems engineer in optimizing the controller. However, the engineer must specify the weighting factors and compare the results with the specified design goals. Often this means that controller synthesis will be an iterative process where the engineer judges the produced optimal controllers through simulation and then adjusts the weighting factors to get a controller more in line with the specified design goals.

1.4 Previous Theory

While our results build on previous work on test signals [11, 33, 31, 10, 27], we do not make identical assumptions. To understand this difference we must briefly summarize the previous work.

1.4.1 Standard Method vs. Incipient Method

We look at two approaches to find a test signal that will detect incipient faults: the standard two model method and the incipient two model method. The results from the two approaches are compared. These approaches are also used in [33].

With the standard two model method we consider finding an auxiliary signal for two fixed models and in order to determine the solution using the incipient two model method we consider a parameterized family of models. We get two models that allow us to find the incipient detection signal after problem reformulation and a limiting process. These models are not identical to the ones found using the standard two model method, but similar mathematical techniques can be used for analysis.

The computed incipient test signal, with scaling, can then be used over a range of parameter values. Since the test signal is designed to work for a range of parameters, the solution of the incipient two model problem can be quite different from the test signal found by the standard two model method which only considers two parameter values. Further, in the two parameter valued case, we get a signal that guarantees detection. In the incipient case since there is some approximation and reformulation involved, we specify a threshold and guarantee detection if some combination of uncertainties exceeds that threshold.

1.4.2 Proper test signal, v

It is important to define what it means for our test signal v to be proper because a proper test signal gives us the ability to distinguish between the normal and faulty system models. Here we specify what it means in the discrete time case. Note that a proper test signal for the continuous time problem has the same definition but slightly different notation.

Assume that we have two discrete linear time invariant models of the form

$$x_{k+1}^i = A_i x_k^i + B_i v_k + M_i \mu_k^i, \quad k = 0, \ldots, K-1 \tag{1.19a}$$
$$y_k^i = C_i x_k^i + N_i \mu_k^i, \quad k = 0, \ldots, K. \tag{1.19b}$$

For the above model, the i indicates model 0 or model 1, where model 0 is considered the normal system and model 1 is considered the faulty system. Also, x_k^i is the state of the system i at a specific time k, v_k is the test signal, and μ_k^i represents the noise/additive uncertainty of the system. The only commonality between the two models is the detection signal, v_k. There are two unknowns, x_0 and μ^i. Let B_i be $n \times m$. Thus, each x_k^i is $n \times 1$ and v_k is $m \times 1$. Suppose each y_k^i is $r \times 1$. Actually, the x_k^i could have different sizes n_i, that is, the state dimension could be different for model 0 and model 1. But we will ignore that for now since this does not occur in the incipient case.

We consider x_0^i to be uncertain or noise. In the approach of [10], it is assumed that the uncertainty in each model is bounded by

$$S_i = (x_0^i)^T Q_i x_0^i + \sum^K (\mu_k^i)^T \mu_k^i < 1 \tag{1.20}$$

where the bound of 1 is taken for convenience. Again, $i = 0$ represents the noise bound for the normal model of a particular system and $i = 1$ corresponds to the noise for a faulty model of the same system. In this thesis we will consider $S_0 + S_1 < 1$. Note that the scaling of M_i easily accommodates other bounds.

Using a static formulation of our problem we define the vectors

$$x^i = \begin{pmatrix} x_1^i \\ \vdots \\ x_K^i \end{pmatrix}, y^i = \begin{pmatrix} y_0^i \\ \vdots \\ y_K^i \end{pmatrix} \mu^i = \begin{pmatrix} x_0^i \\ \mu_0^i \\ \vdots \\ \mu_K^i \end{pmatrix}, v = \begin{pmatrix} v_0 \\ \vdots \\ v_{K-1} \end{pmatrix}.$$

Notice that x^i refers to a collection of time points. This is true for each of the variables with similar notation.

Then (1.19) can be rewritten

$$\mathcal{E}_i x^i = \mathcal{B}_i v + \mathcal{M}_i \mu^i \tag{1.21a}$$

$$y^i = \mathcal{C}_i x^i + \mathcal{N}_i \mu^i \tag{1.21b}$$

where

$$\mathcal{E}_i = \begin{pmatrix} I & 0 & \dots & \dots & 0 \\ -A_i & I & 0 & \dots & \vdots \\ 0 & -A_i & I & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -A_i & I \end{pmatrix}_{K\times K}, \mathcal{M}_i = \begin{pmatrix} A_i & M_i & 0 & \vdots & \vdots & 0 \\ 0 & 0 & M_i & 0 & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & M_i & 0 \end{pmatrix}_{K\times K+2}$$

$\mathcal{B}_i = \mathrm{Diag}(B_i, \dots, B_i)_{K\times K}$,

$$\mathcal{C}_i = \begin{pmatrix} 0 & 0 & \dots & 0 \\ C_i & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & C_i \end{pmatrix}_{K+1\times K},$$

$$\mathcal{N}_i = \begin{pmatrix} C_i & N_i & 0 & \dots & 0 \\ 0 & 0 & N_i & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & N_i \end{pmatrix}_{K+1 \times K+2} .$$

The sizes shown on the block matrices, like $K \times K$ are its block size. That is, for example $\mathcal{E}_i$ is K blocks by K blocks. But $\mathcal{E}_i$ is invertible so we can solve (1.21a) for x^i and substitute into (1.21b) to get

$$\begin{aligned} y^i &= \mathcal{C}_i \left(\mathcal{E}_i^{-1} (\mathcal{B}_i v + \mathcal{M}_i \mu^i) \right) + \mathcal{N}_i \mu^i & (1.22a) \\ &= \mathcal{C}_i \mathcal{E}_i^{-1} \mathcal{B}_i v + (\mathcal{C}_i \mathcal{E}_i^{-1} \mathcal{M}_i + \mathcal{N}_i) \mu^i . & (1.22b) \end{aligned}$$

This shows us that in the discrete time case we may assume that we have two models of the form

$$y^i = X_i v + H_i \mu^i, \quad i = 0, 1, \qquad (1.23)$$

where $X_i = \mathcal{C}_i \mathcal{E}_i^{-1} \mathcal{B}_i, H_i = \mathcal{C}_i \mathcal{E}_i^{-1} \mathcal{M}_i + \mathcal{N}_i$. The form (1.23) will be very convenient in writing our calculations and getting easier to program algorithms. However, since it does not exploit the structure of the matrices involved, it is not the most efficient. Later on the X_i, H_i may come from different problems but this will be a good starting point. The computations based on (1.23) may not be the most efficient for larger problems.

Now, we want to talk about v being proper. A detection signal, v, is proper if it separates the output sets of the two models (i.e., the normal and faulty output sets are disjoint). So suppose that it is possible to get the same output from both models, that is, $y^0 = y^1$. Using (1.23) this says

$$(X_1 - X_0)v = H_0 \mu^0 - H_1 \mu^1 = [H_0, -H_1] \begin{pmatrix} \mu^0 \\ \mu^1 \end{pmatrix} = H\mu. \qquad (1.24)$$

We assume that H is full row rank and we have a noise bound

$$(\mu^0)^T\mu^0 + (\mu^1)^T\mu^1 < 1. \tag{1.25}$$

Now v is proper if $\|\mu\|^2 \geq 1$ for all μ for which (1.24) holds. So it suffices to look at such μ and see if it is greater than or equal to one in norm.

For a fixed v the minimum norm μ satisfying (1.24) is given by

$$H^\dagger(X_1 - X_0)v, \tag{1.26}$$

where $H^\dagger$ is the minimum norm least squares inverse (also called the Moore-Penrose generalized inverse) of H [27]. If $H_{m\times n}$ is full row rank, then the singular value decomposition (SVD) of H is $U[\Sigma\ 0]V^T$, and $H^\dagger = V\begin{pmatrix}\Sigma^{-1}\\ 0\end{pmatrix}U^T$, where U is an $m\times m$ orthogonal matrix, V is an $n\times n$ orthogonal matrix, and since H has full row rank (i.e., the rank is m), $\Sigma = Diag(\sigma_1, \sigma_2, ..., \sigma_m)$ is an $m\times m$ diagonal matrix such that these σ_i's are the nonzero singular values of H. Recall that the σ_i^2's are the eigenvalues of H^*H and the singular vectors for H are the specialized sets of eigenvectors for H^*H.

If $H^\dagger(X_1 - X_0)v$ is less than one in norm, then v is not proper. Now think of picking v. It needs to make (1.26) have norm at least one and also be as small as possible. Take the SVD of $H^\dagger(X_1 - X_0)$. Let σ be the largest singular value of $H^\dagger(X_1 - X_0)$ and let γ be a normalized right singular vector ($\|\gamma\| = 1$) that goes with the singular value σ. Then if we take $v^* = \sigma^{-1}\gamma$ we will have that $H^\dagger(X_1 - X_0)v^*$ has norm one and v^* is proper. In fact, v^* is the smallest test signal that works with the given noise bound (1.25).

Proof. (v^* is the smallest test signal with the given bounds): Suppose σ_1 is the largest singular value of $H^\dagger(X_1 - X_0)$ and let γ_1 be a normalized right singular vector

($\|\gamma_1\| = 1$) that goes with the singular value σ_1. Then if we take $v^* = \sigma_1^{-1}\gamma_1$, we will have that $H^\dagger(X_1 - X_0)v^*$ has norm one and v^* is proper.

Now, take the SVD of $H^\dagger(X_1 - X_0)$ to be given by $U[\Sigma\ 0]V^T$. We want to look at $\|U\Sigma\ V^T v\|^2 = 1$. Since U is an orthogonal matrix, using the two-norm, $\|\Sigma V^T v\|^2 = 1$. Let $w = V^T v$. So we want w to be small and $\|\Sigma w\|^2 = 1$. Notice that since $\Sigma = \text{Diag}(\sigma_i, \ldots)_{n\times n}$ and w is an $n \times 1$ column vector, this gives $\sum \sigma_i^2 w_i^2 = 1$, where $i = 1, 2, ..., n$. Since σ_1 is the largest singular value, take $w_1 = 1/\sigma_1$ and $w_i = 0$ for $i = 2, 3, ..., n$. Then, $\|\Sigma w\|^2 = 1$ so the w_i's we have chosen work. Notice that since $\sum \sigma_i^2 w_i^2 = 1$ and $\sum \sigma_1^2 w_i^2 \geq 1$, this implies $\sigma_1^2\|w\|^2 \geq 1$. Hence, $1/\sigma_1^2 \leq \|w\|^2$, which implies that $1/\sigma_1 \leq \|w\|$ if $\sigma_1 \geq 1$. Therefore, $w = [1/\sigma_1, 0, ..., 0]$, but $w = V^T v$ and V^T is orthogonal by properties of the SVD, so $Vw = v$. This gives the smallest proper test signal v, which we call v^*. □

1.4.3 L^2 norm

L^2 norm of a vector c, $\|c\|$, is defined for a complex vector $c = [c_1, c_2, \ldots, c_n]^T$ by $\|c\| = \sqrt{\bar{c}^T c}$ or $\|c\| = \sqrt{\sum_{i=1}^n |c_i|^2}$, where $|c_i|$ denotes the complex modulus. L^2 norm is the vector norm commonly encountered in vector algebra and vector operations such as the dot product. It is also referred to as the two-norm or the Euclidean norm for vectors.

We use the two (or euclidean) norm for the total noise in our models. The work done in [10] used the max (or infinity) norm on the noise in the models, therefore we are solving a related, but different problem. Recall that for any finite vector $c, \|c\|_\infty \leq \|c\|_2 \leq \|c\|_1$. See [27]. Using the two norm makes our inner minimum problem easier to deal with.

1.4.4 β Bounds and Maximum

Now we discuss how the parameter β which occurs in [32] is related to our development of the noise measure we use later in this thesis. We assume that β is a number such that $0 \leq \beta \leq 1$ and we use it in the definition of the bound on the proper test signal v.

Here we outline the work done in [32] to define parameter β and how we use it to determine proper test signals. In the multi-model framework of the auxiliary signal design problem as presented in [10], the system models for the normal and failed systems of the continuous time case are taken to be the following two models over the test period $[0, T]$:

$$x_i' = A_i x_i + B_i v + M_i \nu_i \tag{1.27a}$$

$$y = C_i x_i + D_i v + N_i \nu_i \tag{1.27b}$$

for $i = 0$ and 1. v is the auxiliary signal input, y is the measured output, x_i's are the states, and ν_i's are noise inputs. $A_i, B_i, C_i, D_i, M_i, N_i$ are matrices of appropriate dimensions. The $N_i's$ are assumed to have full row rank. Model (1.27) for $i = 0$ represents the normal system and $i = 1$ represents the failed system.

The models considered in [10] are in fact more general and allow for model uncertainty, but here we only consider the additive noise situation. The assumption on the noise inputs and the uncertainty on initial conditions are

$$S_v^i(x_i(0), \nu_i) \equiv x_i(0)^T P_{i,0}^{-1} x_i(0) + \int_0^T \|\nu_i\|^2 \quad dt < 1, \quad i = 0, 1. \tag{1.28}$$

$P_{i,0}$ are positive-definite matrices specifying the amount of uncertainty in the initial condition. $S_v^i(x_i(0), \nu_i)$ measures the size of the disturbances with respect to which the detection must be robust.

Recall that the $L^2[0, T]$ vector function v is a proper auxiliary signal if its application implies that we are always able to distinguish the two candidate models based on the observation of y.

In the two model case, noise is measured using a max norm. Recall the assumption is that there is a normal model and a faulty model, which we refer to as model 0 and model 1, respectively. Assuming equal outputs the signal v is not proper if $\max\{\|S_0\|, \|S_1\|\} < 1$. We want to find a minimal proper test signal.

The problem formulation for finding v is

$$\min_v \|v\| | \min_{\mu_i, x_i(0)} (\max\{\|S_0\|, \|S_1\|\}) > 1,$$

where the $\mu_i, x_i(0)$ represent the noise from each respective model. However, $\max\{\|S_0\|, \|S_1\|\}$ is difficult to solve since noise is measured as a maximum noise. Therefore, we replace it with an expression that is simpler to work with. The parameter β is introduced to simplify the problem formulation and to get an LQR problem with $\max_{0\leq\beta\leq 1}$ on the outside of our expression.

Using the fact that

$$\max\{\|x\|, \|z\|\} = \max_{0\leq\beta\leq 1} \beta\|x\| + (1-\beta)\|z\|$$

and assuming $y_0 = y_1$ we can rewrite the problem formulation in the following manner

$$\min_v \|v\| | \min_{\mu_i, x_i(0)} \max_{0\leq\beta\leq 1} \beta\|S_0\| + (1-\beta)\|S_1\| \geq 1. \tag{1.29}$$

We can switch the order of the minimum and maximum and can now rewrite (1.29) in the following form:

$$\min_v \|v\| | \max_{0\leq\beta\leq 1} \min_{\mu_i, x_i(0)} \beta\|S_0\| + (1-\beta)\|S_1\| \geq 1. \tag{1.30}$$

Now we have an LQR control problem for a given β. In fact, Equation (1.30) is an LQR problem for each β. We assume $\beta = \frac{1}{2}$, which gives the expression $\frac{1}{2}\|S_0\| + \frac{1}{2}\|S_1\| \geq 1$.

Definition 6. *The auxiliary signal v is not proper if there exists x_0, x_1, ν_0, ν_1 and y satisfying (1.27) and (1.28) both for $i = 0$ and $i = 1$. The auxiliary signal v is called proper otherwise.*

If V denotes the set of proper auxiliary signals v,

$$\gamma^* = \inf_{v \in V} \|v\|^2 = \inf_{v \in V} \int_0^T |v|^2 \quad dt, \tag{1.31}$$

is a lower bound on the energy of proper auxiliary signals. $\lambda^* = 1/\gamma^*$ is called the separability index. A greater separability index implies the existence of a lower energy proper auxiliary signal. The separability index is zero when there is no proper auxiliary signal.

It is important to explain how to compute the separability index. The characterization of the set S_v can be simplified by noting that the output y can be eliminated from the constraints (1.27a)-(1.27b), $i = 0, 1$, by subtracting (1.27b) for $i = 1$ from (1.27b) for $i = 0$, giving

$$0 = C_0 x_0 - C_1 x_1 + N_0 \nu_0 - N_1 \nu_1 + (D_0 - D_1) v. \tag{1.32}$$

To simplify the notations, let

$$x = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \nu = \begin{pmatrix} \nu_0 \\ \nu_1 \end{pmatrix},$$

and

$$A = \begin{pmatrix} A_0 & 0 \\ 0 & A_1 \end{pmatrix}, B = \begin{pmatrix} B_0 \\ B_1 \end{pmatrix}, C = \begin{pmatrix} C_0 & -C_1 \end{pmatrix},$$

$$D = \begin{pmatrix} D_0 & -D_1 \end{pmatrix}, M = \begin{pmatrix} M_0 & 0 \\ 0 & M_1 \end{pmatrix}, N = \begin{pmatrix} N_0 & -N_1 \end{pmatrix}.$$

Then the constraints (1.27a), $i = 0, 1$, and (1.32) can be expressed as

$$x_i' = Ax + Bv + M\nu \qquad (1.33a)$$

$$0 = Cx + Dv + N\nu. \qquad (1.33b)$$

The separability index λ^* can then be computed as follows

$$\lambda^* = \max_{0 \leq \beta \leq 1} \lambda_\beta \qquad (1.34)$$

where

$$\lambda_\beta = \max_{v \neq 0} \frac{\phi_\beta}{\|v\|^2} \qquad (1.35)$$

subject to the constraints in (1.33) where

$$\phi_\beta = \inf_{\nu, x} x(0)^T P_\beta^{-1} x(0) + \int_0^T \nu^T J_\beta \nu \quad dt \qquad (1.36)$$

and

$$J_\beta = \begin{pmatrix} \beta I & 0 \\ 0 & (1-\beta)I \end{pmatrix}, P_\beta = \begin{pmatrix} \frac{1}{\beta}P_{0,0} & 0 \\ 0 & \frac{1}{1-\beta}P_{1,0} \end{pmatrix}.$$

Theorem 1.1 2. *λ_β is the infimum over all λ's for which the Riccati equation*

$$P' = (A - S_{\lambda,\beta}R_{\lambda,\beta}^{-1}C)P + P(A - S_{\lambda,\beta}R_{\lambda,\beta}^{-1}C)^T - PC^TR_{\lambda,\beta}^{-1}CP + Q_{,\beta} - S_{\lambda,\beta}R_{\lambda,\beta}^{-1}S_{\lambda,\beta}^T \tag{1.37}$$

with $P(0) = P_\beta$ has a non-negative definite solution P on $[0,T]$ where the following equation holds:

$$\begin{pmatrix} Q_{\lambda,\beta} & S_{\lambda,\beta} \\ S_{\lambda,\beta}^T & R_{\lambda,\beta} \end{pmatrix} = \begin{pmatrix} M & B \\ N & D \end{pmatrix} \begin{pmatrix} J_\beta & 0 \\ 0 & -\lambda I \end{pmatrix}^{-1} \begin{pmatrix} M & B \\ N & D \end{pmatrix}^T. \tag{1.38}$$

Once we have computed β^*, an optimal value of β in (1.34), and the separability index λ^*, we can proceed with the construction of minimal energy proper auxiliary signal v.

Lemma 1. *The two-point boundary value system:*

$$\frac{d}{dt}\begin{pmatrix} x \\ \zeta \end{pmatrix} = \begin{pmatrix} \Omega_{11} & \Omega_{12} \\ \Omega_{21} & \Omega_{22} \end{pmatrix} \begin{pmatrix} x \\ \zeta \end{pmatrix} \tag{1.39}$$

with the following boundary conditions

$$\begin{aligned} x(0) &= P_{\beta^*}\zeta(0) & (1.40)\\ \zeta(\tau) &= 0 & (1.41)\end{aligned}$$

where

$$\begin{aligned} \Omega_{11} &= -\Omega_{22}^T \\ &= A - S_{\lambda^*,\beta^*}R_{\lambda^*,\beta^*}^{-1}C & (1.42)\\ \Omega_{12} &= Q_{\lambda^*,\beta^*}R_{\lambda^*,\beta^*}^{-1}S_{\lambda^*,\beta^*}^T & (1.43)\\ \Omega_{21} &= C^T R_{\lambda^*,\beta^*}^{-1}C & (1.44)\end{aligned}$$

is well-posed for $\tau < T$ but it is not well-posed, i.e., it has non-trivial solutions (x, ζ) for $\tau = T$.

Theorem 1.1 3. *Let (x, ζ) be a non-trivial solution of the two-point boundary value system (1.39). Then an optimal auxiliary signal is*

$$v^* = \alpha((-B + S_{\lambda^*,\beta^*}R_{\lambda^*,\beta^*}^{-1}D)^T\zeta + D^T R_{\lambda^*,\beta^*}^{-1}Cx) \tag{1.45}$$

where α is a normalization constant such that $\|v^*\| = 1/\sqrt{\lambda^*}$. Details on implementation issues and numerical algorithms in Scilab for computing the optimal auxiliary *signal can be found in [10].*

In order to solve the optimal control problem it is essential to use our definition of a proper detection signal v. Note that for the detection signal v to be not proper, (1.27) must hold as well as (1.28). We can rewrite (1.28) as

$$\max\{x_0(0)^T P_{0,0}^{-1} x_0(0) + \int_0^T \|\nu_0\|^2 dt, \quad x_1(0)^T P_{1,0}^{-1} x_1(0) + \int_0^T \|\nu_1\|^2 dt\} < 1. \tag{1.46}$$

This expression can also be written as

$$\max_{0\le\beta\le1}\{\beta(x_0(0)^T P_{0,0}^{-1} x_0(0) + \int_0^T \|\nu_0\|^2) dt + (1-\beta)(x_1(0)^T P_{1,0}^{-1} x_1(0) + \int_0^T \|\nu_1\|^2) dt\} < 1. \tag{1.47}$$

In other words, $\max(S_0, S_1) = \max_{0\le\beta\le1} \beta S_0 + (1-\beta)S_1$. In the previous work done on incipient faults, like [22], the maximum noise of the two models is considered, i.e. $\max(S_0, S_1) < 1$. Thus, we obtain a useful characterization of not proper detection signals based on the noise bound S_i.

Lemma 2. *The detection signal v is not proper if and only if*

$$\inf \max_{0\le\beta\le1}\{\beta(x_0(0)^T P_{0,0}^{-1} x_0(0) + \int_0^T \|\nu_0\|^2) dt + (1-\beta)(x_1(0)^T P_{1,0}^{-1} x_1(0) + \int_0^T \|\nu_1\|^2) dt\} < 1 \tag{1.48}$$

where the infimum is taken over $(x_i, \nu_i, y) \in L^2$ subject to (1.27), $i = 0, 1$.

The above characterization is useful because the algorithm we develop will compute the minimum energy proper detection signal by finding the detection signal of smallest norm that does not satisfy (1.48) [22].

In this thesis we work with the sum of the noise from both the normal and faulty models, $S_0 + S_1 < 1$. We find the smallest noise that works. Therefore, we use the least noise that makes the output sets from the normal and faulty models disjoint.

This noise allows the two output sets to be as close as possible without having a common output y.

If the bound is $\max(S_0, S_1) < 1$, then it is possible to run separate filters on both models which was the approach used in [11, 31]. However, in our algorithms we deal with one less level of optimization so our use of the test signal is slightly different. We suppose that we have a set of incipient faults parameterized by Ω and given by a vector $\delta\theta$. We assume that the noise bound for each model separately is

$$S_i = x_i(0)^T x_i(0) + \int_0^{\omega} \nu_i^T \nu_i + \mu_i^T \mu_i \, dt < \delta^2.$$

Then we compute the test signal v^* for a bound of 1 and let $\hat{v} = \delta\sqrt{2}v^*$. Using this test signal we have that if we get the same output from both the normal and any of the faulty models, then one of the models must have noise S_i larger than δ^2. We then run the standard filter on just the normal model. More details on this method can be found in chapter four.

In general, the β parameter requires an iteration to solve for it. However, in [31] it is seen that when the single parameter incipient problem is reformulated so that the machinery of [10] can be used, we find that $\beta = \frac{1}{2}$. For the additive uncertainty case studied here, the use of a fixed $\beta \in [0, 1]$ does not affect the existence of a proper test signal. Consequently, the use of a fixed β may result in obtaining a suboptimal proper test signal as discussed in [11].

From [11] we get some terminology that pertains to the β which is chosen. Proper and minimal proper refer to the approach that uses a β search or an iteration to select β. The terms subproper and minimal subproper refer to the method that uses $\beta = \frac{1}{2}$. Since a subproper bound admits a larger set of uncertainties, if v is a subproper signal it is also proper.

The algorithms may take different forms, but there is a maximization over the parameter β where $0 \leq \beta \leq 1$. There is also an alternative problem, which resembles a robust control problem, where there is no β. We consider the case where $\beta = \frac{1}{2}$.

1.4.5 Additive Uncertainty Case

This thesis focuses primarily on additive uncertainty. Consequently, the case with only additive uncertainty differs from the model uncertainty case in several important ways. First, if v is proper (or subproper), then taking cv for $c > 1$ is still proper (or subproper). In the case with model uncertainty it is possible for the A_i to grow in size as c increases [11]. Since we consider models which include additive uncertainty, we can establish an upper bound. However, due to our setup of the multi-parameter incipient fault problem, a type of model uncertainty also arises in our model formulation. See Sections 3.2.3 and 4.8 for a detailed discussion about how model uncertainty is taken into account in our problem.

We have linear models of the form (4.1), (4.2) and we have the additive uncertainty case. Let us assume v^* and $\underline{v}^*$ be the minimal proper and minimal subproper test signals, respectively. Then, our cost function J satisfies $J(v^*) \leq J(\underline{v}^*) \leq 2J(v^*)$. The proof can be found in [11].

Chapter 2

Discrete Time Case With One Fault

As introduced in the previous chapter, our goal is to find an optimal input signal that detects incipient faults in linear uncertain systems using a multi-model approach. We construct two models. One model represents the true system, which we call the normal model and one which represents the faulty version of the same system, which we call the faulty model. The normal model depends on a parameter vector θ and the faulty model depends on parameter θ plus drift parameter vector $\delta\theta$, where θ is fixed and the drift parameter $\delta\theta$ may vary within a specified threshold. In this chapter, our focus will be on finding an optimal detection signal for the discrete time problem with only one incipient fault so that θ is a scalar.

2.1 General Model for Active Fault Detection

This chapter highlights the methods used in previous research to detect a single incipient fault in discrete linear time invariant models. In particular, we look at some examples of finding a minimal test signal v that satisfies given conditions and detects the incipient fault as early as possible. This lays the groundwork for my research problem, which extends this concept to the case where there are multiple incipient faults occurring simultaneously. These results are taken from [10, 30, 31, 32].

We present the first active multi-parameter incipient fault detection algorithm. Our goal is to use an active approach to detect incipient faults in linear systems where there is more than one drifting parameter. We determine an auxiliary signal v that is used to detect small parameter variations in linear time invariant uncertain systems.

Multi-model fault detection means that we have two or more possible models for a given system. We consider a normal and a faulty model which corresponds to a particular system. Then we determine a proper test signal that will allow us to distinguish between the two models over a finite time interval. For the discrete time case we consider the test period $[0, K]$. While there are other possible test periods, we will restrict our discussion to a finite interval.

We assume that we have two linear time invariant models of the form (1.19) with the variables defined the same as in chapter one. We begin by looking at a discrete linear time invariant model, where A_i,B_i,C_i,M_i, and N_i do not depend on k. However, with appropriate modifications a similar method can be applied to study the linear time varying model.

Note that in general the test signal v behaves in the following manner. If something is done early in the time interval it can have an effect on the input v (i.e., the control in our problem) that lasts for a while, but something done towards the end of the time interval has little chance to have an effect on the control. As a result, the test signal v usually is at or close to zero at the end of the time interval. If there is a weight on the initial value, then the largest part of the test signal v tends to be early. If there is no weight on the initial value, then v tends to be small at the start. If the initial condition is unknown, then the control waits for the effect of the initial condition to get out of the way and then acts.

The following problem is an example of calculating the optimal test signal using the two model method.

Example 1. *We illustrate a specific example in the form (1.19) where we find the optimal detection signal* v^* *given* A_i, B_i, C_i, M_i, N_i *for* $i = 0, 1$ *and* K. *We use the machinery developed by Campbell and Nikoukhah to solve this problem. Let*

$$A_0 = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, A_1 = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}, B_0 = B_1 = \begin{pmatrix} 1 & 2 \\ 1 & 3 \end{pmatrix},$$

$$C_0 = C_1 = \begin{pmatrix} 0 & 1 \end{pmatrix}, M_0 = M_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}, N_0 = N_1 = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix},$$

and $K = 7$. *The code for this example can be found in the appendix.*

The solution is in Figure 2.1. Since v_k *is two dimensional,* v_k^* *is two dimensional for each* k *with components* v_0, v_1 *at each time step. The resulting plot is a graph showing two discrete time functions. The line segment connecting values are given to improve readability.*

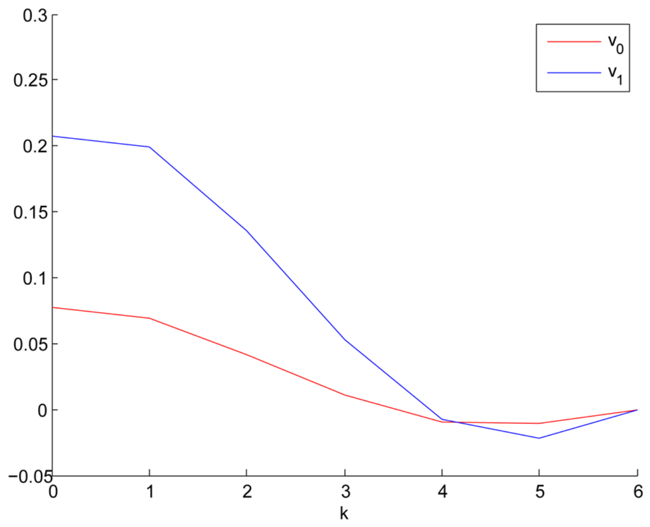

Figure 2.1: Test Signal v^* with $K = 7$ for Example 1

2.2 Incipient Problem with One Fault

Before developing our multi-fault results we discuss the one fault case, which was previously studied [14, 15, 31, 32]. The work done in [31, 32] give a solution to the problem of auxiliary signal design using an on-line implementation of the fault detector. These studies and other similar work provide insights into studying incipient faults and helped motivate my thesis topic which extends this work to examine the case where more than one fault occurs concurrently in linear models.

For the incipient problem we start with the normal model which depends only on parameter θ

$$x_{k+1} = A(\theta)x_k + B(\theta)v_k + M\mu_k, \quad k = 0, \dots, K-1 \tag{2.1a}$$

$$y_k = C(\theta)x_k + N\mu_k, \quad k = 0, \dots, K, \tag{2.1b}$$

and the faulty model which also includes parameter drift $\delta\theta$

$$\hat{x}_{k+1} = A(\theta + \delta\theta)\hat{x}_k + B(\theta + \delta\theta)v_k + M\hat{\mu}_k, \quad k = 0, \dots, K-1 \tag{2.2a}$$

$$\hat{y}_k = C(\theta + \delta\theta)\hat{x}_k + N\hat{\mu}_k, \quad k = 0, \dots, K. \tag{2.2b}$$

There are multiple ways to approach this problem and we study the first one.

Method I: Write (2.2) in the form (1.23) and develop the algorithms at this level. This produces the optimal test signal for a specific $\delta\theta$.

Method II: Take (2.1) and (2.2) and proceed to get a system. Much of it has been done in previous work which focuses on studying incipient faults. Here we use linearization with respect to θ and get a test signal that scales with the desired threshold on $\delta\theta$. Then we write this system as some generalization of (1.23) and solve it.

2.2.1 Method I

Method I has already been done. We focus on Method II.

2.2.2 Method II

Suppose that we have our two models already in the form

$$y^0 = X(\theta)v + H(\theta)\mu^0 \tag{2.3}$$

$$y^1 = X(\theta+\delta\theta)v + H(\theta+\delta\theta)\mu^1. \tag{2.4}$$

Setting $y^1 = y^0$ we get

$$0 = X(\theta+\delta\theta)v + H(\theta+\delta\theta)\mu^1 - X(\theta)v - H(\theta)\mu^0$$

or

$$0 = X'(\theta)\delta\theta v + H(\theta)\mu^1 - H(\theta)\mu^0 + O(\delta\theta).$$

Setting

$$w = \delta\theta v,$$

and dropping the other $O(\delta\theta)$ terms we get

$$X_\theta(\theta)w = [H(\theta), -H(\theta)]\mu, \tag{2.5}$$

when $X_\theta = X'(\theta)$. We can now proceed as before in Section 1.4.2. We take w^* to be $\frac{1}{\sigma}\gamma$ where σ is the largest singular value of $[H(\theta), -H(\theta)]^\dagger X_\theta(\theta)$ and γ is a normalized right singular vector that goes with σ.

If (2.3), (2.4) come from (2.1) and (2.2) and we use the notation of (1.21), then

$$X(\theta) = \mathcal{C}(\theta)\mathcal{E}^{-1}(\theta)\mathcal{B}(\theta), \quad H(\theta) = \mathcal{C}(\theta)\mathcal{E}^{-1}(\theta)\mathcal{M}(\theta) + \mathcal{N}. \tag{2.6}$$

Using the fact that the derivative of Z^{-1} is $-Z^{-1}Z'Z^{-1}$, we have that we can get $X'(\theta)$ in terms of the simpler matrices.

In fact,

$$X' = \mathcal{C}'\mathcal{E}^{-1}\mathcal{B} - \mathcal{C}\mathcal{E}^{-1}\mathcal{E}'\mathcal{E}^{-1}\mathcal{B} + \mathcal{C}\mathcal{E}^{-1}\mathcal{B}'$$

where all terms are evaluated at θ.

Example 2. *The following example illustrates Method II in finding the optimal incipient test signal. Suppose*

$$X(\theta) = \begin{pmatrix} 1+\theta & \theta^2 \\ 2+3\theta & 2\theta^2 \\ \theta^2+\theta & 1 \end{pmatrix}, H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \theta_0 = 0.$$

Given $X(\theta), X'(\theta), H(\theta), \theta_0, \delta\theta$, we compute w^* as in (2.5). For $\delta\theta = 0.1, 0.5, 1.0$, and 10.0 we compute the incipient detection signal v^*_{inc} and the two model test signal

v^*. w^* is calculated in order to find $v^*_{inc} = \frac{1}{\delta\theta}w^*$. We compute v^* by looking at the models with θ and $\theta + \delta\theta$ and finding the test signal, which we refer to as the two model method (Method I). The two model (standard) test signal v^* and the incipient test signal v^*_{inc} are compared by plotting them on the same graph.

For larger $\delta\theta$ the value of v^* decreases and v^*_{inc} just scales with different values of $\delta\theta$. We find that it is possible that v^* will change shape as $\delta\theta$ gets larger. Further, as we decrease the value of $\delta\theta$ (the small error term) for the one parameter case, the two lines that represent the standard test signal v^* and the incipient test signal v^*_{inc} are closer to being the same which was expected.

If there are no $O(\delta\theta)$, we have a model linear in $\delta\theta$ and $v^* = v^*_{inc}$. We take $v^*_{inc} = \frac{1}{\delta\theta}w^*$ (where the shape of w^* is independent of $\delta\theta$). If the problem is linear in θ, then v^* and v^*_{inc} should be close in value. However, if the problem is nonlinear, then v^* and v^*_{inc} will not be close in value. Further, if $X(\theta)$ is affine in θ, meaning $X(\theta) = X_0 + \delta\theta X_1$ for constant matrices X_0, X_1, then we expect v^* and v^*_{inc} to be the same for all $\delta\theta$. If $X(\theta)$ is nonlinear in θ, we expect v^* and v^*_{inc} to be close for small $\delta\theta$, but they may be different for large $\delta\theta$. See below for plots.

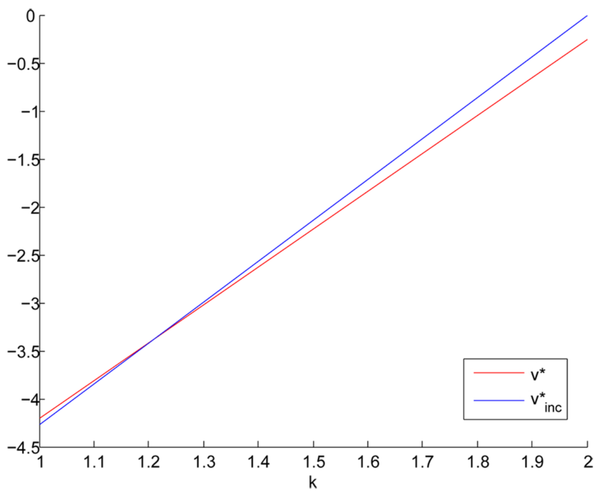

Figure 2.2: v^* and v^*_{inc} when $\delta\theta = 0.1$ for Example 2

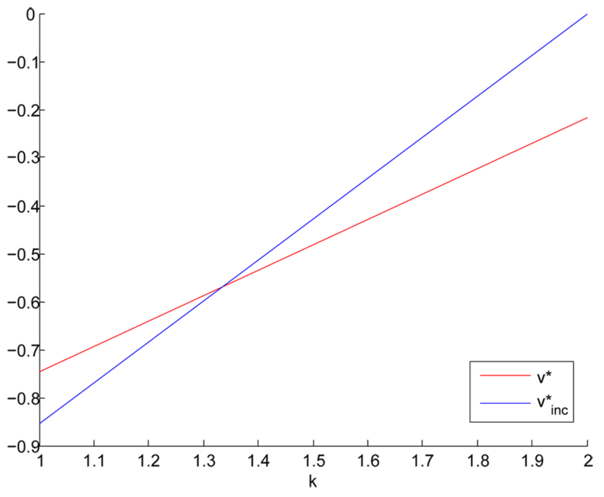

Figure 2.3: v^* and v^*_{inc} when $\delta\theta = 0.5$ for Example 2

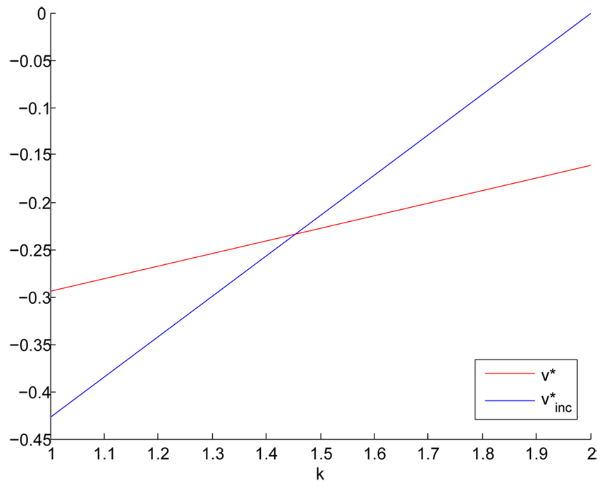

Figure 2.4: v^* and v^*_{inc} when $\delta\theta = 1$ for Example 2

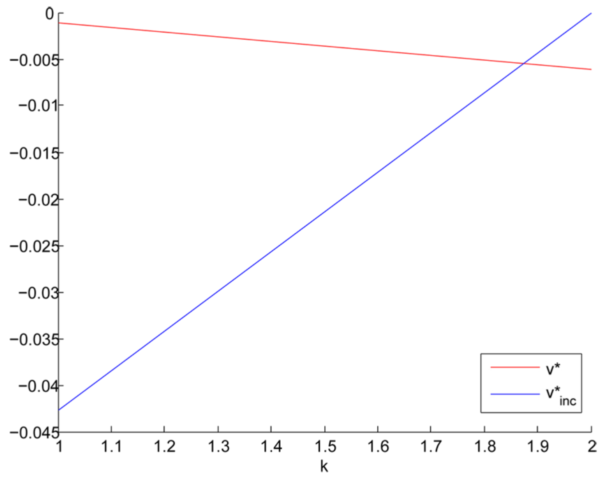

Figure 2.5: v^* and v^*_{inc} when $\delta\theta = 10$ for Example 2

2.2.3 Getting into the form (2.3) and (2.4)

Setting $y_k = \hat{y}_k$ and combining (2.1) and (2.2) we get the constraints which state what must happen to generate the same output.

$$\begin{aligned} x_{k+1} &= A(\theta)x_k + B(\theta)v_k + M\mu_k, \quad k = 0, \ldots, K-1 && (2.7a)\\ \hat{x}_{k+1} &= A(\theta+\delta\theta)\hat{x}_k + B(\theta+\delta\theta)v_k + M\hat{\mu}_k, \quad k = 0, \ldots, K-1 && (2.7b)\\ 0 &= -C(\theta)x_k - N\mu_k + (C(\theta+\delta\theta)\hat{x}_k + N\hat{\mu}_k), \quad k = 0, \ldots, K. && (2.7c) \end{aligned}$$

For convenience we let $Q = Q(\theta)$ and $Q_\theta = Q_\theta(\theta)$ for a matrix function Q. We let

$$\bar{x}_k = \hat{x}_k - x_k,\ \bar{z} = \delta\theta\hat{x}_k, \bar{v}_k = \delta\theta v_k$$

and (2.7) becomes

$$\begin{aligned} \bar{x}_{k+1} &= A\bar{x}_k + A_\theta\bar{z}_k + B_\theta\hat{v}_k + M\hat{\mu}_k - M\mu_k, \quad k = 0, \ldots, K-1 && (2.8a)\\ \bar{z}_{k+1} &= A\bar{z}_k + B\hat{v}_k, \quad k = 0, \ldots, K-1 && (2.8b)\\ 0 &= C\bar{x}_k + C_\theta\bar{z}_k - N\mu_k + N\hat{\mu}_k, \quad k = 0, \ldots, K. && (2.8c) \end{aligned}$$

We now rewrite (2.8) in the same way that we rewrote (1.19) into (1.24). Let

$$x = \begin{pmatrix} \bar{x}_1 \\ \bar{z}_1 \\ \vdots \\ \bar{x}_K \\ \bar{z}_K \end{pmatrix},\ \mu = \begin{pmatrix} \bar{x}_0 \\ \bar{z}_0 \\ \mu_0 \\ \hat{\mu}_0 \\ \vdots \\ \mu_K \\ \hat{\mu}_K \end{pmatrix},\ \hat{v} = \begin{pmatrix} \hat{v}_0 \\ \vdots \\ \hat{v}_{2K-1} \end{pmatrix},\ y = \begin{pmatrix} y_0 \\ \hat{y}_0 \\ \vdots \\ y_K \\ \hat{y}_K \end{pmatrix}$$

Then, we get

$$\mathcal{E} = \left(\begin{array}{cc|cc|cc|cc} I & 0 & 0 & \dots & \dots & \dots & 0 & 0 \\ 0 & I & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ \hline -A & -A_\theta & I & 0 & \ddots & \ddots & \vdots & \vdots \\ 0 & -A & 0 & I & \ddots & \ddots & \vdots & \vdots \\ \hline \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ \hline \vdots & \vdots & \ddots & \ddots & -A & -A_\theta & I & 0 \\ 0 & \dots & \vdots & \vdots & 0 & -A & 0 & I \end{array}\right)_{2K\times 2K},$$

$$\mathcal{M}_i = \begin{pmatrix} A_i & M_i & 0 & \vdots & \vdots & 0 \\ 0 & A_i & M_i & 0 & \vdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & M_i & 0 \end{pmatrix}_{2K\times 2K+4},$$

$\mathcal{B}_i = \mathrm{Diag}(B_i, \dots, B_i)_{2K\times 2K}$,

$$\mathcal{C}_i = \begin{pmatrix} 0 & 0 & \dots & 0 \\ C_i & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & C_i \end{pmatrix}_{2K+2\times 2K},$$

$$\mathcal{N}_i = \begin{pmatrix} C_i & N_i & 0 & \dots & 0 \\ 0 & C_i & N_i & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & N_i \end{pmatrix}_{2K+2\times 2K+4}.$$

$\mathcal{E}_i$ is invertible so we can solve and write the output as

$$y^i = \mathcal{C}_i\left(\mathcal{E}_i^{-1}(\mathcal{B}_i\hat{v} + \mathcal{M}_i\mu^i)\right) + \mathcal{N}_i\mu^i \tag{2.9a}$$

$$= \mathcal{C}_i\mathcal{E}_i^{-1}\mathcal{B}_i\hat{v} + (\mathcal{C}_i\mathcal{E}_i^{-1}\mathcal{M}_i + \mathcal{N}_i)\mu^i. \tag{2.9b}$$

This shows us that in the discrete time case we may assume that we have two models of the form

$$y^i = X_i\hat{v} + H_i\mu^i, \quad i = 0, 1. \tag{2.10}$$

where $X_i = \mathcal{C}_i\mathcal{E}_i^{-1}\mathcal{B}_i, H_i = \mathcal{C}_i\mathcal{E}_i^{-1}\mathcal{M}_i + \mathcal{N}_i$. The solution to (2.10) gives an optimal test signal $\hat{v}$.

Chapter 3

Discrete Time Case With Multiple Faults

In studying the multiple incipient fault problem, we examine two cases: discrete time and continuous time. This chapter focuses on the discrete time case. It provides an introduction to the novelty of this thesis. Here we develop an algorithm to find a solution to the multi-parameter incipient fault problem, specifically for the two fault case and the three fault case. These results are presented in two papers written based on the first results of this dissertation [16, 17].

3.1 Incipient Problem with Two Faults

Fault identification is important when a fault evolves slowly since the associated loss of performance in a system can go unnoticed for some time. We begin by highlighting work that was recently published which gives an example of how to detect two incipient faults in a practical system. In [45] an observer based method of estimating the magnitudes of slowly developing faults in heating, ventilating, and air conditioning (HVAC) equipment was studied. This paper provides a method for estimating the sizes of two types of incipient faults in a cooling coil subsystem. The two types of incipient faults considered are drift in the electronics controlling the position of the valve actuator and air-side fouling of the coil. This work shows how to successfully identify two types of incipient faults in a cooling coil system of an air-handling unit. Note that nondetection for this particular case study could result in a reduction in the cooling capacity of the coil and an increase in the energy consumption of the fan.

We now begin our development by introducing the multi-parameter fault problem and presenting an active multi-fault incipient fault detection algorithm for more than two faults. We begin by studying the case where two faults occur concurrently in a system. Then, we focus on the problem where there are three faults occurring at the same time in a given system. This chapter includes examples for each case. The model and algorithms described in this chapter will allow for more than three incipient faults to occur simultaneously.

We start with the simplest case which is the static one and we assume a model of the form

$$y = X(\theta, \phi)v + H(\theta, \phi)\mu^0. \tag{3.1}$$

We let $\psi = (\theta, \phi)$ and $\delta\psi = \begin{pmatrix} \delta\theta \\ \delta\phi \end{pmatrix}$. The faulty model is

$$y = X(\psi + \delta\psi)v + H(\psi + \delta\psi)\mu^1. \tag{3.2}$$

Taking the differences of (3.1) and (3.2) and dropping the error terms except for those with v we get the following linearized system

$$X_\theta(\psi)\delta\theta v + X_\phi(\psi)\delta\phi v = H(\psi)\mu^1 - H(\psi)\mu^0. \tag{3.3}$$

We could let $w_1 = \delta\theta v$ and $w_2 = \delta\phi v$ and get a problem with two controls in it. But this is deceptive since w_1 and w_2 are multiples of each other.

We may look at this point further later but let us consider alternatives right now that can get us closer to the one parameter case. One is that we suppose that

$$\delta\theta = \alpha_1 \delta\kappa, \quad \delta\phi = \alpha_2 \delta\kappa. \tag{3.4}$$

Next, we assume that $\alpha = (\alpha_1, \alpha_2)$ lies in some set Ω and $\delta\kappa$ is a fixed value. Then equation (3.3) becomes

$$\left(H^{\dagger}(X_\theta \alpha_1 + X_\phi \alpha_2)\right) w = \mu \tag{3.5}$$

where $w = \delta\kappa v$.

We want to find a w that will allow us to detect incipient faults for any $\alpha \in \Omega$. One way to proceed is to fix w with $\|w\|$=1. Then we minimize the norm of $H^{\dagger}(X_\theta \alpha_1 + X_\phi \alpha_2)w$ over α. We then maximize over the w to do the best possible. We may then need to adjust the size of the test signal. This leads us to the following expression. Let α^*, w^* be solutions of

$$\zeta = \max_{\|w\|=1} \min_{\alpha \in \Omega} \|H^{\dagger}(X_\theta \alpha_1 + X_\phi \alpha_2)w\|. \tag{3.6}$$

Then the test signal would be $\frac{1}{\zeta} w^*$. The optimization problem (3.6) can be challenging. Note that (3.6) is equivalent to

$$\zeta^2 = \max_{\|w\|=1} \min_{\alpha \in \Omega} \|H^{\dagger}(X_\theta \alpha_1 + X_\phi \alpha_2)w\|^2. \tag{3.7}$$

There are a variety of different choices to make for Ω.

One option is to suppose that $\delta\theta + \delta\phi = \delta\kappa$. This says that we are taking a certain level of perturbation but not worrying about what variable causes it. But then $\alpha_1 + \alpha_2 = 1$. In other words, given an interval Π we have $\Omega = (\beta, 1-\beta) : \beta \in \Pi$. There should be some bounds on how big Π is. One possibility is $\Pi = [0, 1]$. This is sort of the convex hull of the two one parameter cases. That is, the case with just ϕ changing and the case with just θ changing. We begin by looking at this particular case. Two examples of the incipient fault problem with two faults follow.

Example 3. *As an illustration of how different the solution can be in the two parameter versus the one parameter incipient case, we first consider a simple illustrative academic example where the calculations are straightforward. Suppose*

$$y = \begin{pmatrix} 2+\theta & -1+3\phi \\ 2 & 1 \end{pmatrix} v + H_0\mu_0. \tag{3.8}$$

We take our model of the form found in (1.23), where $H_0 = I$ and I is a 2×2 identity matrix. We assume that $\alpha_1 = \epsilon, \alpha_2 = 1-\epsilon$ and $0 \le \epsilon \le 1$. Without loss of generality, we can take the nominal values of θ, ϕ to be zero. Then (3.5) is

$$\begin{pmatrix} \frac{1}{2}I \\ -\frac{1}{2}I \end{pmatrix} \begin{pmatrix} \epsilon & 3(1-\epsilon) \\ 0 & 0 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \mu. \tag{3.9}$$

For this example we can easily solve the optimization problem (3.6).

Before prceeding we note that if ϕ is identically zero, then this is the same as taking $\epsilon = 0$ and we have the optimal test signal would be

$$\tilde{w} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{3.10}$$

According to (3.6) we must solve

$$\zeta = \max_{\|w\|=1} \min_{0 \le \epsilon \le 1} \frac{1}{2} \|\epsilon w_1 + 3(1-\epsilon) w_2\|. \tag{3.11}$$

We have $w_1^2 + w_2^2 = 1$. The inner min is over ϵ. We are then going to max over the w. Note that if $\|w\| = 1$, then for a particular value of w_1, w_2, there are four feasible w with the same norm by using $\pm w_1$ and $\pm w_2$. We get a larger value of the min if the signs of the w_1, w_2 are the same. Then the inner min will be the smaller of $w_1, 3w_2$. But when we take the max over $\|w\| = 1$ this occurs when $w_1 = 3w_2$ and we get that $w_1 = \frac{3}{\sqrt{10}}, w_2 = \frac{1}{\sqrt{10}}$. Now, it must be rescaled so that the right hand side of (3.9) has norm one. Thus

$$\frac{1}{\zeta}\sqrt{\frac{20}{9}} \begin{pmatrix} \frac{3}{\sqrt{10}} \\ \frac{1}{\sqrt{10}} \end{pmatrix} = \sqrt{2} \begin{pmatrix} 1 \\ \frac{1}{3} \end{pmatrix}. \tag{3.12}$$

We note that if we had a different set Ω, say one defined by $\alpha_1^2 + \alpha_2^2 = 1$ with (α_1, α_2) in the first quadrant, then the calculation shows that we get the same shape. That is, the normalized vector w^*, since again the argument is a balancing one. However, we have to use a slightly larger multiple for a given parameter variation threshold. That is because $\alpha_1^2 + \alpha_2^2 = 1$ allows for slightly larger uncertainty for (α_1, α_2) away from the origin, $\alpha_1 = 0$ and $\alpha_2 = 0$ axis as shown in Figure 3.1.

In Example 3 the dependence on the parameters θ, ϕ is linear. This means that there are no errors in the reformulation. The next academic example has the param-

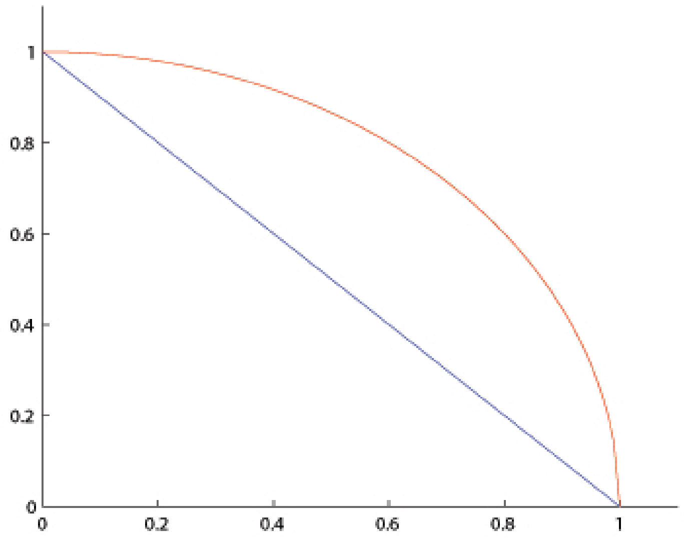

Figure 3.1: Example 3 $\alpha_1 + \alpha_2 = 1$ (red) and $\alpha_1^2 + \alpha_2^2 = 1$ (blue) in the first orthant.

eters entering nonlinearly.

Example 4. *Let*

$$y = \begin{pmatrix} 1+\theta & \theta^2 \\ 2+3\phi-2\theta & 2\phi^2 \\ \theta\phi+1 & 1 \end{pmatrix} v + H_0\mu_0, \tag{3.13}$$

where H_0 is a 3×3 identity matrix. We assume that $\alpha_1 = \epsilon, \alpha_2 = 1-\epsilon$ and $0 \le \epsilon \le 1$. We take the nominal values of θ, ϕ to be zero. For a given $\epsilon, \delta\kappa$, let $v^*(\epsilon, \delta\kappa)$ be the optimal test signal found with the two model method and v^*_{inc} be the optimal incipient test signal such that $v^*_{inc} = \frac{1}{\delta\kappa} w^*$.

For more complex problems we can move directly to a general optimization code, but for this example it is interesting to examine things more carefully. If we parameterize w by $w = [\cos x, \sin x]^T$, then the graph of the value of the inner min in

Equation (3.7) is given in Figure 3.2. We see that the maximum of this minimum occurs at x=0 (and also $\pi, 2\pi$*, etc.; it is periodic). Therefore,* $w = [1, 0]^T$.

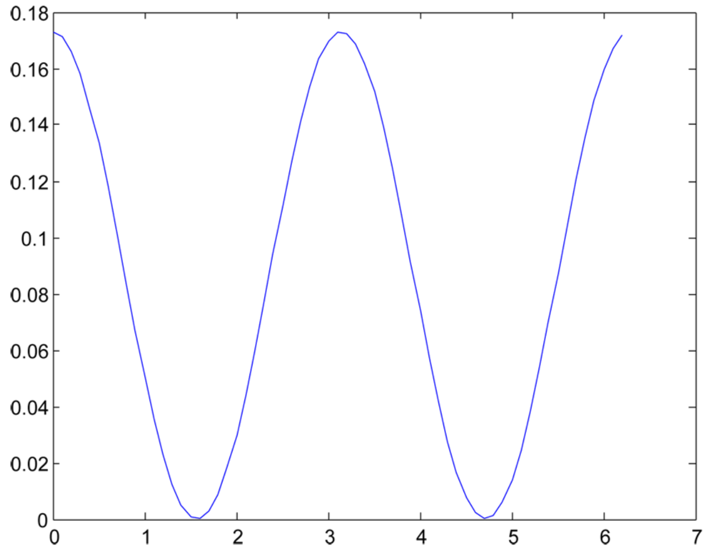

Figure 3.2: Example 4 graph of the inner min in (3.7) as a function of x

Since the incipient test signal must cover a range of problems, we expect it to be larger than just one test signal for a special case. We also expect that the incipient test signal will more closely approximate the usual test signal for smaller $\delta\kappa$*. Figure 3.3 and Figure 3.4 illustrate this for* $\delta\kappa = 0.1$ *and* $\delta\kappa = 1$*. In each figure the test signals are drawn as two dimensional vectors. The red one is* v^*_{inc}*. The black ones are the standard two model signals for that* $\delta\kappa$ *and* $\epsilon = 0, 0.3, 0.5, 0.7, 0.9, 1.0$*. We see that* v^*_{inc} *is closer to the two model test signals when* $\delta\kappa = 0.1$ *in Figure 3.3 than when* $\delta\kappa = 1.0$ *in Figure 3.4.*

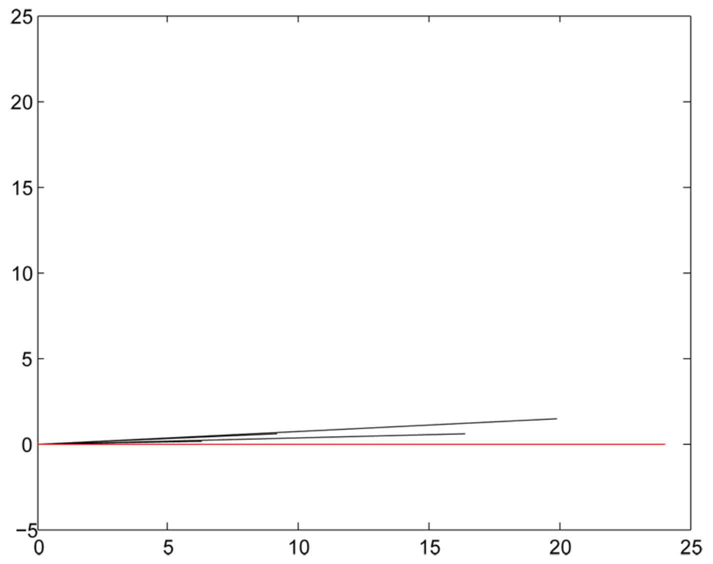

Figure 3.3: Example 4 v^*_{inc} and the standard test signals for $\delta\kappa = 0.1$ and $\epsilon = 0, 0.3, 0.5, 0.7, 0.9, 1.0$

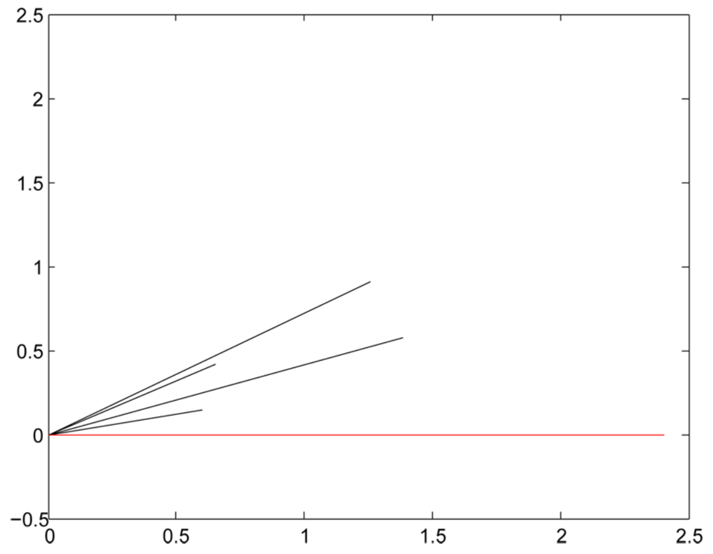

Figure 3.4: Example 4 v^*_{inc} and the standard test signals for $\delta\kappa = 1$ and $\epsilon = 0, 0.3, 0.5, 0.7, 0.9, 1.0$

3.2 More Than Two Simultaneous Incipient Faults

This section focuses on the case where there are r concurrent incipient faults in a system. For the incipient problem, we start with the normal model

$$\begin{aligned} x_{k+1} &= A(\theta)x_k + B(\theta)v_k + M\mu_k, \quad k \in [0:K-1] && (3.14a)\\ y_k &= C(\theta)x_k + N\mu_k, \quad k \in [0:K] && (3.14b) \end{aligned}$$

where $\theta = (\theta_1, \ldots, \theta_r)^T$ is a vector of r parameters. Thus we allow r simultaneous faults. Let $\delta\theta = (\delta\theta_1, \ldots, \delta\theta_r)^T$ be the vector of parameter variations. Model (3.14) with parameter drift is

$$\begin{aligned} \hat{x}_{k+1} &= A(\theta+\delta\theta)\hat{x}_k + B(\theta+\delta\theta)v_k + M\hat{\mu}_k, \quad k \in [0:K-1] && (3.15a)\\ \hat{y}_k &= C(\theta+\delta\theta)\hat{x}_k + N\hat{\mu}_k, \quad k \in [0:K]. && (3.15b) \end{aligned}$$

Again using the notation of 3.1 we can rewrite (3.14) and (3.15) in the static formulation as

$$\begin{aligned} y^0 &= X(\theta)v + H(\theta)\mu^0 && (3.16)\\ y^1 &= X(\theta+\delta\theta)v + H(\theta+\delta\theta)\mu^1. && (3.17) \end{aligned}$$

Our goal is to determine the shape of a good test signal independent of the value of $\delta\theta$ and then be able to scale the test signal depending on the threshold with which we wish to guarantee detection of the fault.

Setting $y^1 = y^0$ we get

$$0 = X(\theta+\delta\theta)v + H(\theta+\delta\theta)\mu^1 - X(\theta)v - H(\theta)\mu^0$$

or

$$0 = \sum_{j=1}^{r} X_{\theta_j}(\theta)\delta\theta_j v + H(\theta)\mu^1 - H(\theta)\mu^0 + O(\delta\theta)$$

where $X_{\theta j} = dX/d\theta_j$. Dropping the O terms except for those with v, we get after a little calculation that

$$\sum_{j=1}^{r} X_{\theta_j}(\theta)\delta\theta_j v = H(\theta)\mu^1 - H(\theta)\mu^0. \tag{3.18}$$

We could let $w_j = \delta\theta_j v$ and we could get a problem with r vector valued controls in it. However, if we did this then all of the w_i are multiples of each other. Again, we address this by using a formulation that allows us to use the results from the one parameter incipient case as a subproblem. Since we know that the w_j are multiples of the same test signal, we suppose that

$$\delta\theta_j = \alpha_j \delta\kappa, \quad j = 1, \ldots, r. \tag{3.19}$$

We assume that $\alpha = (\alpha_1, \ldots, \alpha_r)^T$ lies in some closed, bounded subset Ω of $\Re^r$ for a fixed $\delta\kappa$. For detection to be possible, we need that $0 \notin \Omega$. Then the equation (3.18) for the smallest μ that results in equal outputs for a given α becomes

$$\bar{H}^{\dagger}(\sum_{j=1}^{r} \alpha_j X_{\theta_j}(\theta))w = \mu \tag{3.20}$$

where $w = \delta\kappa v$ and $\bar{H} = [H(\theta), -H(\theta)]$.

We want to find a test signal w that can be used to detect incipient faults for all $\alpha \in \Omega$. As we did with the two parameter case, one way to proceed is fix w with $\|w\| = 1$ and then minimize the norm of $\bar{H}^{\dagger}(\sum_{j=1}^{r} \alpha_j X_{\theta_j}(\theta))w$ over $\alpha \in \Omega$. That is, we minimize over α. Then we maximize over the w. We may then need to adjust the size of w. This leads to the following expression. Let α^*, w^* be solutions of

$$\zeta = \max_{\|w\|=1} \min_{\alpha\in\Omega} \|\bar{H}^\dagger \sum_{j=1}^{r}(\alpha_j X_{\theta_j}(\theta)w)\|. \tag{3.21}$$

Then, the test signal shape would be $\frac{1}{\zeta}w^*$.

There are a variety of different choices to make for Ω depending on the particular application. One option is to suppose that $\sum_{j=1}^{r}\delta\theta_j = \delta\kappa$. This says that we are taking a certain level of perturbation, but not worrying about what variable causes it. But then

$$\sum_{j=1}^{r}\alpha_j = 1. \tag{3.22}$$

Another possibility is when

$$\underline{\alpha_j} \le \alpha_j \le \overline{\alpha_j} \tag{3.23}$$

and at least one interval $[\underline{\alpha_j}, \overline{\alpha_j}]$ does not contain zero. This formulation is attractive for several reasons. For one, it leads to box constraints which many optimizer packages are built to consider. Secondly, it has a natural interpretation in terms of each parameter value falling into a certain interval.

3.2.1 Solving (3.21)

At first glance the optimization problem (3.21) may appear difficult to solve. In fact, it can be quickly solved for small to medium sized problems as we will now explain. We consider the inner minimization problem first. Suppose that we have a given w such that $\|w\| = 1$. Let $b_j = \bar{H}^\dagger X_{\theta_j}(\theta)w$. Then the square of the norm inside (3.21) is

$$\|\sum_{j=1}^{r}\alpha_j b_j\|^2 = \sum_{j=1}^{r}\sum_{i=1}^{r} c_{i,j}\alpha_i\alpha_j \tag{3.24}$$

where $c_{i,i} = b_i^T b_j$. This is a positive semi-definite quadratic form over Ω. There are a number of fast routines for minimizing quadratic functions. If the region Ω is given by (3.22) or by (3.23) with $r = 2$, then we get the minimization of a quadratic form over box constraints.

The inner minimization is thus of a smooth function. For the outer optimization in (3.21) the cost may not be smooth but there is the constraint $\|w\| = 1$. One could just use this constraint but many optimization packages prefer simpler box type constraints. Suppose the w is K-dimensional. Using the usual parameterization of the K-sphere we can replace $\|w\| = 1$ by box constraints. The parameterization is

$$w_1 = \cos(\gamma_1) \tag{3.25a}$$
$$w_2 = \sin(\gamma_1)\cos(\gamma_2) \tag{3.25b}$$
$$w_3 = \sin(\gamma_1)\sin(\gamma_2)\cos(\gamma_3) \tag{3.25c}$$
$$\vdots \tag{3.25d}$$
$$w_{K-1} = \sin(\gamma_1)\ldots\sin(\gamma_{K-2})\cos(\gamma_{K-1}) \tag{3.25e}$$
$$w_K = \sin(\gamma_1)\ldots\sin(\gamma_{K-1}), \tag{3.25f}$$

where $0 \le \gamma_i \le \pi$ for $i \in [1, K-2]$ and $0 \le \gamma_{K-1} \le 2\pi$. Note that there is symmetry to the optimization problem so that if w is optimal, then so is $-w$. Thus, we can reduce the size of the optimization domain by taking

$$0 \le \gamma_i \le \pi, \quad i \in [1, K-1]$$

as parameterizing w. The inner minimization is r dimensional, which is the total number of incipient faults found in our system. Note that typically r is not a large integer. The outer maximization is K dimensional.

3.2.2 Using the Test Signal

Suppose that we have our test signal v and an observed output $\tilde{y}$. To use the test signal we compute the norm squared of the smallest noise that makes $\tilde{y}$ consistent with model zero (i.e., the normal model). That is,

$$\rho = \|H_0^{\dagger}(\theta)(\tilde{y} - X_0(\theta)v)\|^2. \tag{3.26}$$

If $\rho < 1$ (which we consider small), then we do not detect the fault at the indicated level. If $\rho > 1$, then we conclude there is an incipient fault. We can do this since we assume the noise in both models is bounded by 1. That is,

$$S_i = (x_0^i)^T Q_i x_0^i + \sum^{K} (\mu_k^i)^T \mu_k^i < 1 \tag{3.27}$$

and proper v is designed to produce disjoint output sets.

If the model equations are affine in $\delta\theta$, then there is guaranteed fault detection. The only conservatism is in perhaps using a slightly larger test signal than is minimally necessary due to our use of the two norm for the combined norm.

Two problems in which we solve for the incipient test signal when there are three concurrent incipient faults follow. If the $\delta\theta$ enter in a nonlinear manner, then the approximation error could possibly increase with the size of the parameter change. This is similar to the incipient one fault case. Note that we are seeking to detect incipient faults early or when they are still small. We start with an academic example.

Example 5. *Let*

$$y = \begin{pmatrix} 2+\theta_1 & -1+3\theta_2 \\ 2 & 1+\theta_3 \end{pmatrix} v + H\mu.$$

We take the nominal value θ equal to a zero vector and $H = I$, where I is a 2×2

identity matrix. Using the notation of (3.23) we assume Ω *is of the form*

$$0.5 \leq \alpha_1 \leq 1.0, \tag{3.28a}$$

$$0.1 \leq \alpha_2 \leq 0.5, \tag{3.28b}$$

$$0 \leq \alpha_3 \leq 0.1. \tag{3.28c}$$

We know that if w *is optimal, then so is* $-w$*. Therefore, it suffices to parameterize half of the* w *sphere. Using the sphere parameterization (3.25), we have that*

$$q(w,\alpha) = \begin{pmatrix} \frac{1}{\sqrt{2}} I \\ -\frac{1}{\sqrt{2}} I \end{pmatrix} \begin{pmatrix} \alpha_1 & 3\alpha_2 \\ 0 & \alpha_3 \end{pmatrix} \begin{pmatrix} \cos(\gamma) \\ \sin(\gamma) \end{pmatrix}$$

where $0 \leq \gamma \leq \pi$ *and we solve*

$$-\min_{0\leq\gamma\leq\pi} -\min_{\alpha\in\Omega} \|q(w,\alpha)\|^2.$$

Figure 3.5 is a graph of the inner minimum as a function of γ*. The outer optimization gives the maximum of* γ *as 0.17 at* $x = 0.5407$*. Note that the plot for the three parameter case (Figure 3.5) is highly asymmetrical unlike the inner min for the two parameter case (Figure 3.2).*

Figure 3.6 compares the incipient test signal (in red) and the standard two model test signals (in black) for several values of $\alpha \in \Omega$*.*

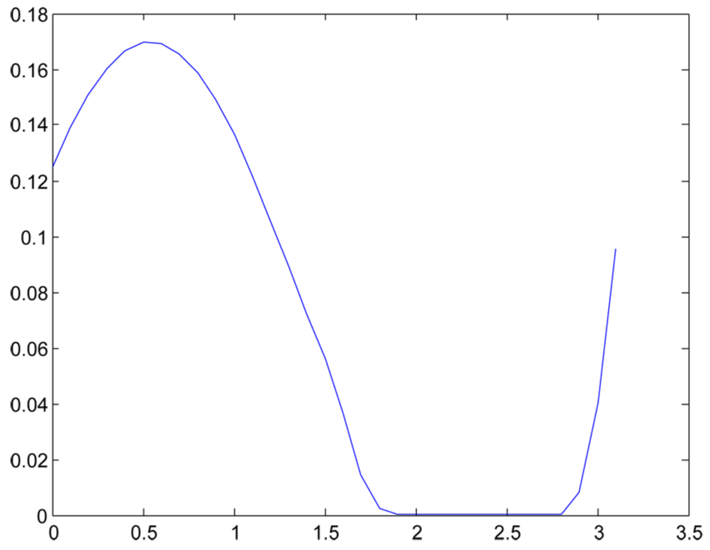

Figure 3.5: Example 5 inner min as a function of γ

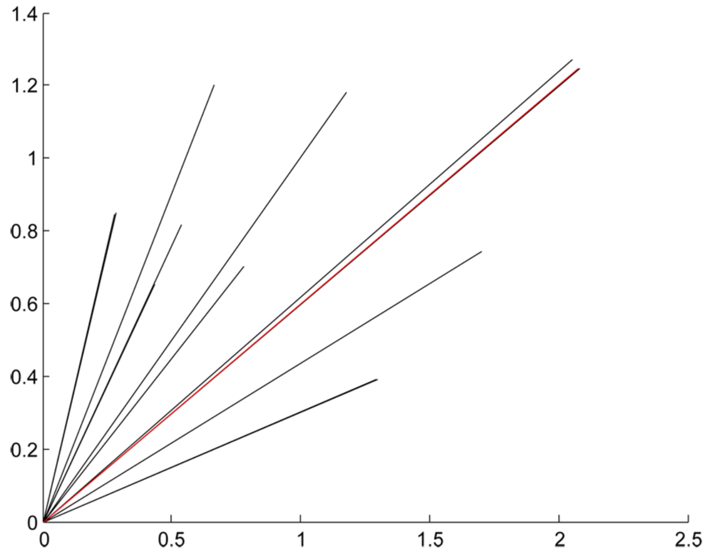

Figure 3.6: Incipient test signal (red) and two model signals for several values of α (black) for Example 5

Example 6. *Suppose*

$$\begin{aligned} x_{k+1} &= \begin{pmatrix} 1 & h \\ -hsm & -hmd \end{pmatrix} x_k + \begin{pmatrix} 0 \\ hm \end{pmatrix} v + \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \mu_k, \quad k = 0, \dots, K-1 \\ y_k &= \begin{pmatrix} 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \mu k, \quad k = 0, \dots, K. \end{aligned}$$

To illustrate the use of this approach we consider the discrete system above. Note that $M_0 = M_1$ and $N_0 = N_1$. This problem is actually the Euler approximation of a mechanical system with damping coefficient d, spring constant s, mass $\frac{1}{m}$, and h as the step size. The output y_k is the position of the mass. Note that the parameters enter nonlinearly.

We assume that the normal values are $\theta_1 = m = 2, \theta_2 = s = 12, \theta_3 = d = 0.2$. This is a lightly damped mechanical system with eigenvalues $0.4600 \pm 0.8176i$ and modulus 0.9381. We assume that we want a test signal that is guaranteed to determine a fault has occurred where the fault condition is $\theta_1 + \delta\theta_1 \in [0.9, 1.1], \theta_2 + \delta\theta_2 \in [8, 10]$, and $\theta_3 + \delta\theta_3 \in [0.4, 0.6]$. Therefore, $\delta\theta_1 \in [-1.1, -0.9], \delta\theta_2 \in [-4, -2]$, and $\delta\theta_3 \in [0.2, 0.4]$. Using the nominal values and the incipient fault condition, we can conclude that there has been a change or there is uncertainty in the mass, the spring has weakened, and the friction has increased due to say a break down in lubricant. We can take $\delta\kappa = 1$ and then $\alpha_i = \delta\theta_i$. For this example, we take $h = 0.2$ and $K = 5$. Our set Ω is now a *rectangular box that does not include the origin. With these parameter values we get*

$$v^*_{inc} = \begin{pmatrix} 18.2136 \\ -8.6901 \\ -2.8236 \\ 0.0485 \\ 0 \end{pmatrix}. \tag{3.30}$$

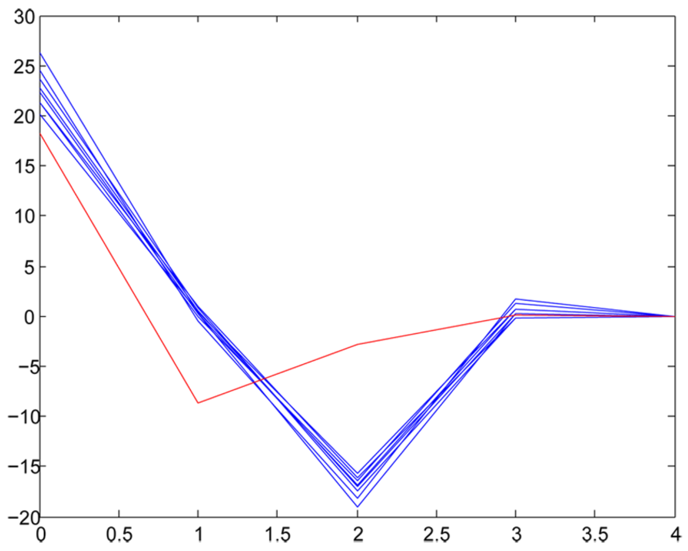

Figure 3.7: Incipient test signal (red) and two model signals for several α (blue) for Example 6

Figure 3.7 shows the incipient test signal in red and the two model test signals for the values of α *at the corners of* Ω*. Since the noise is symmetric about the origin we have that* $-v$ *is proper if* v *is proper. To better compare the incipient and two model test signals we have plotted the negative of the incipient test signal in Figure 3.7. It should be noted that the parameterization (3.25) can create artificial local minimum* *when using the numerical optimization code. For example, the parameterization has* $0 \le \gamma_1 \le \pi$*. However, if the actual optimum appears at say* $\frac{-1}{6}\pi$ *and* $\frac{5}{6}\pi$*, and we use* $0 \le \gamma_1 \le \pi$*, then depending on the initial guess chosen, the optimizer might go to* $\gamma_1 = 0$*. In fact, this happens with this particular example. This was easily overcome by giving wider bounds on the* γ_1 *when the optimizer is called.*

3.2.3 Conclusions for the Discrete Time Case

When considering incipient faults the usual assumption that there is only one fault happening at a given time is no longer always appropriate [31]. In this chapter, we have begun the examination of an active approach for incipient fault detection when there are more than two faults in the discrete time case. As with some of the earlier results [15, 31, 32, 33] we specify a threshold and guarantee detection if some combination of the uncertainties exceeds that threshold.

For the discrete time case, v^*_{inc} covers a range of problems so we expect it to be larger than just one test signal for a special case. For smaller $\delta\kappa, v^*_{\text{inc}}$ is closer to the solution of the two model test signals.

The problem to be solved is a min max problem. In principle this can be solved by software. However, more efficient algorithms are needed for higher state dimensions and longer time horizons. Developing these more efficient algorithms is under investigation.

In previous work done by Nikoukhah and Campbell, finding a test signal in the presence of model uncertainty was carried out with a reformulation that led to a new optimization problem which added the uncertainty to the noise [31]. The approach of this section has provided an alternative approach to including some types of model uncertainty when the faults are incipient. In the use of the ideas of this section some parameters are treated as faults and some of them as model uncertainties. The difference is that if the jth parameter is uncertain, then we have $0 \in [\underline{\alpha_j}, \overline{\alpha_j}]$ and if the jth parameter is incipient, then $0 \notin [\underline{\alpha_j}, \overline{\alpha_j}]$. This approach to test signal design in the presence of model uncertainty is a topic for further study.

Chapter 4

Continuous Time Case

This chapter focuses on the continuous time case. The model is similar to the discrete time case, but the approach for solving it is different. We begin by constructing the model problem and using the information from the discrete time case that is relevant. Then we proceed by adding new details necessary to find a solution for the incipient detection signal for continuous time models. The results from this section are highlighted in [18].

4.1 Continuous Time Two Model Version

For the continuous time case, we want to recapture the essence of what was done in the discrete time case. We begin in a similar way, but now the model includes two ordinary differential equations. We examine this model by setting up a linear quadratic regulator (LQR) problem.

Here, like in the discrete time case, we start with a multi-model approach on a finite time interval $[0, \omega]$. We assume we have two ode models of the following form:

$$x_i' = A_i x_i + B_i v + M_i \mu_i \tag{4.1a}$$

$$y_i = C_i x_i + N_i \mu_i, \tag{4.1b}$$

which is the continuous analog of (1.19). For the above model, the i indicates model 0 or model 1, where model 0 is considered the normal system and model 1 is considered the faulty system. Again x_i is the state, v is the test signal, and μ_i represents the additive uncertainty in the model. The commonality between the two models is the detection signal, v.

The initial state $x_i(0)$ and noise μ_i are unknowns in each model. Let B_i be $n \times m$. Thus, each x_i is $n \times 1$ and v is $m \times 1$. Suppose each y_i is $r \times 1$. Note that the x_i could have different sizes n_i. That is, the state dimension could be different for model 0 and model 1, but we will ignore that for now since this does not occur in the incipient case.

We consider x_0^i to be additive uncertainty. In the approach of [10], it is assumed that the uncertainty in each model is bounded by

$$S_i(x_i(0), \mu_i) = x_i(0)^T x_i(0) + \int_0^\omega \mu_i^T(t)\mu_i(t)\,dt < 1, \quad i = 0, 1 \tag{4.2}$$

where (4.2) is actually an inner product norm on $(x_i(0), \mu_i)$ and the bound of 1 is taken for convenience. Scaling of M_i easily accommodates other bounds. Note that bound (4.2) is the continuous analog of (1.20).

4.1.1 Proper v

Once the noise is bounded we can consider the outputs of the models. For now suppose $A_i(v)$ is the output set of model i for $i = 0$ and 1. The detection signal v is called a proper test signal if $A_0(v) \cap A_1(v) = \emptyset$. This means that there is no output

y that can come from both the normal model and the faulty model. Our goal is to get a proper signal so that we can distinguish between the two models of the system like we did in the discrete time case.

Suppose $L(f)$ is the solution of the following ode with the given initial condition

$$z' = A_i z + f, \quad z(0) = 0 \tag{4.3}$$

and define it as

$$L(f) = \int_0^t e^{(t-\tau)A_i} f(\tau)\, d\tau. \tag{4.4}$$

Then, the solutions to the set of equations in (4.1) are

$$x_i = L_i(B_i v) + L_i(M_i \mu_i) + e^{A_i t} x_i(0) \tag{4.5a}$$

$$y_i = (C_i L_i B_i)v + (C_i L_i M_i + N_i)\mu_i + C_i e^{A_i t} x_i(0) \tag{4.5b}$$

where $(C_i L_i B_i)v = \hat{L}_i(v)$ is a fixed function that is linearly dependent on the test signal v. In fact, $\hat{L}_i$ is a linear transformation of v and $C_i e^{A_i t} x_i(0) + (C_i L_i M_i + N_i)\mu_i$ is a linear transformation from $(x_i(0), \mu_i)$ that maps into y-space. Notice that the set of $(x_i(0), \mu_i)$ satisfying our noise bound (4.2) is convex and bounded by 1 in the S_i norm. Therefore, when a continuous linear transformation is applied to this bounded and convex set, the new set is also bounded and convex.

Recall how we distinguished between the normal and faulty models in the discrete time case. We found a proper detection signal, which separated the output sets of the two models, i.e. the normal and faulty output sets were disjoint. In order to show this, here again we assume that we can get the same output from both models, that is, $y_0 = y_1$. However, if this were true, then the noise bound would be violated since proper v ensures that the output sets are distinguishable.

Recall our assumption is that the noise in each model is bounded and we assume that (4.2) holds. Hence, if $(x_0(0), x_1(0), \mu_0, \mu_1)$ satisfies the models and v is a proper detection signal, but an output y is still in both output sets, then it must be that

$$\max(S_0(x_0(0), \mu_0), S_1(x_1(0), \mu_1)) \geq 1. \tag{4.6}$$

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. Thus, it is enough to ensure it is true for the minimum of them. In this problem $\min\max = \max\min$ so we get the same result even when we switch the order. Therefore, we can take

$$\min_{x_i, \mu_i} \max(S_0(x_0(0), \mu_0), S_1(x_1(0), \mu_1)) \geq 1. \tag{4.7}$$

This is the classical setup.

As stated in chapter one we use $S_0 + S_1 < 1$ as our noise bound. This noise bound has the advantage that the perturbations are measured by a single inner product norm. This type of constraint makes the problem look more like a robust control problem [11]. This allows the use of robust control theory which is a method for measuring the performance changes of a control system with changing system parameters. In developing the continuous time case the previous work used β and a switch in the order of the maximum and minimum to get a problem that resembles an LQR problem.

4.2 Multi-parameter Incipient Fault Setup

There are several ways to generalize our previous results from the incipient fault problem in the discrete time case. There will be a number of changes as we begin to study the continuous time case. However, we want to recapture the essence of what we have already done:

- For a given detection signal v and parameters, we want to find the worst fault.

This is a minimum problem to find the smallest noise that works. Using an LQR with a fixed end $z(0)$, we then minimize over the endpoint $z(0)$ [Inner Problem];

- With v fixed, we find the minimum over the parameters;
- Then we maximize over v.

This approach can be done two ways. It can be done with a scalable incipient fault level like the $\delta\kappa$ we had before, which is based on linearizations. It can also be done for a certain set of $\delta\theta$. In the discrete time case we did the former. Here we do the latter so there are no linearizations involved in our work for the continuous time case.

The normal model used to represent the behavior of a system subject to incipient faults can be expressed as follows:

$$x' = A(\theta)x + B(\theta)v + M\mu \tag{4.8a}$$

$$y = C(\theta)x + N\nu. \tag{4.8b}$$

Notice the change in the notation. Before we had a big noise vector μ that appeared in both equations (i.e., the ode and output of the ode). Now, we have a different noise in each equation. This actually allows for more noise, but more importantly, we now have N as an invertible matrix which will be useful later. This is the more general case since it allows all output channels to have noise.

For the continuous time problem we are on the time interval $[0, \omega]$. $\theta = (\theta_1, \dots, \theta_r)^T$ is a vector containing r parameters used for either the incipient parameters or model uncertainty or both. Thus, we allow r simultaneous faults. Let $\delta\theta = (\delta\theta_1, \dots, \delta\theta_r)^T$ be the vector of r parameter variations. We refer to $\delta\theta$ as a slow drift in the parameter over time. The model with parameter drift $\delta\theta$, which we call the faulty model is

$$\hat{x}' = A(\theta+\delta\theta)\hat{x} + B(\theta+\delta\theta)v + M\hat{\mu} \tag{4.9a}$$

$$\hat{y} = C(\theta+\delta\theta)\hat{x} + N\hat{\nu}. \tag{4.9b}$$

One goal is to determine the shape of a good test signal independent of the value of $\delta\theta$ and then be able to scale the test signal depending on the threshold with which we wish to guarantee detection of the fault.

For proper v we assume that the noise measure is

$$S = x(0)^T x(0) + \hat{x}(0)^T \hat{x}(0) + \int_0^\omega \nu^T\nu + \mu^T\mu + \hat{\nu}^T\hat{\nu} + \hat{\mu}^T\hat{\mu}\, dt < 1. \tag{4.10}$$

So we write this as

$$S(x(0), \hat{x}(0), \nu, \hat{\nu}, \mu, \hat{\mu}) < 1. \tag{4.11}$$

For a given v we derive the smallest noise that results in equal outputs for all $\delta\theta$. That is, we determine our cost function $J = \min S$ assuming the output from the normal and faulty system models are equal. We start by minimizing over the free endpoints $x(0), \hat{x}(0)$. With this formulation we will be able to use the results from the one parameter incipient case as a subproblem for the multi-parameter case. Note that we do not need to calculate μ and ν explicitly. The noises are a mix of uncertainty, disturbances, and noise. Equation (4.10) merely states that we assume some bound on this noise. Given this bound we construct a test signal that will work as long as our assumption on the noise level is correct.

Note that (4.11) is not identical to the noise bound used in earlier work such as [10]. There is a maximum over separate norm bounds for each model which is used to compare the overall noise bound. Then, this max is replaced by an equivalent max using the parameter β. Recall that in the incipient approach of [31] the parameter $\beta = \frac{1}{2}$. With $\beta = \frac{1}{2}$, the noise bound of [31] is twice that of (4.11). Thus, (4.11) is related to the earlier work and simplifies our algorithms since we deal with one less level of optimization. However, it will require some other modifications such as in how we use the test signal to perform a test. This will be discussed further later.

Recall that a test signal v is proper if we use v and we get the same output from both (4.8) and (4.9) then we must have a violation of our noise bound (4.11). Meaning, with proper v it is impossible to get the same output from both models. We want the smallest such v. A variety of norms can be used depending on the application [10]. Here we use the L^2 norm of the detection signal v. It is important to note that if we start with a v and then compute the smallest noise that produces the same output from both models, say the amount of noise is σ, then due to the linearity of the systems we study $\frac{v}{\sqrt{\sigma}}$ would be proper.

The closer the faulty model is to the nonfaulty model, the larger the test signal must be. However, practical considerations will limit how large a test signal can be, but this is similar to many other results in control theory. Controllability means we can steer from point a to point b, but if the time interval is too short then the control may not be practical.

4.2.1 Multi-Parameter Case

We assume the possible perturbations $\delta\theta$ lie in a set $\tilde{\Omega}$ that does not include the origin. In particular, recall that in chapter three the $w_j = \delta\theta_j v$ are multiples of the same test signal so we suppose that

$$\delta\theta_j = \alpha_j \delta\kappa, \quad j = 1, \ldots, r. \tag{4.12}$$

We assume that $\alpha = (\alpha_1, \ldots, \alpha_r)^T$ lies in some closed and bounded subset Ω of R^r for a fixed $\delta\kappa$. For detection to be guaranteed, we need that $0 \notin \Omega$. Since Ω is closed and bounded and in a finite dimensional space, if $0 \in \Omega$, then 0 must be a positive distance from Ω. Computational algorithms usually also require that Ω is the closure of its interior.

Now we need to derive $S(v, \delta\theta)$, which is the smallest noise that results in equal outputs for a given v. This means we need to solve the following problem:

$$\min S(x(0), \hat{x}(0), \nu, \hat{\nu}, \mu, \hat{\mu}) \tag{4.13}$$

subject to

$$\begin{aligned} x' &= A(\theta)x + B(\theta)v + M\mu && \text{(4.14a)} \\ \hat{x}' &= A(\theta+\delta\theta)\hat{x} + B(\theta+\delta\theta)v + M\hat{\mu} && \text{(4.14b)} \\ 0 &= C(\theta)x + N\nu - C(\theta+\delta\theta)\hat{x} - N\hat{\nu} && \text{(4.14c)} \end{aligned}$$

with v known and fixed. Also, the $\delta\theta_i's$ are bounded. We get (4.14c) by taking the output from the normal and faulty models and equating them. Also, note that since N is invertible, we can use (4.14c) to eliminate ν or $\hat{\nu}$ from both (4.13) and (4.14). Then, (4.13) and (4.14) become a type of LQR problem. Once the smallest noise $S(v, \delta\theta)$ is determined we need to minimize over $\delta\theta$ and maximize over v. Finally, we get a value S_*, which is used in our calculation of the optimal proper test signal v^*.

4.3 Continuous Time Inner Min LQR Problem

In this section we discuss how to solve the inner minimization problem. The algorithm for finding the test signal and the model used to solve the optimization problem are highlighted. We build on the results found here in Section 4.4.

The following is the setup for the inner minimization problem. We shall give both the formulation of a general control problem and our particular control problem. We start with the general control problem.

4.3.1 General Control Problem

Before we get to the specific control problem we will solve, we first review a formal derivation of the necessary conditions for a general optimal control problem. The problem formulation and solution highlighted in this section come from [26]. The general form of the model is

$$x' = f(x, u, t), \quad t \geq t_0 \tag{4.15}$$

with t_0 fixed, which means the system model depends on the state $x(t) \in \Re^n$, control input $u(t) \in \Re^m$, and time t. With this system let us associate the performance index

$$J(u) \;=\; \phi(x(T), T) + \int_{t_0}^{T} L(x, u, t)dt, \tag{4.16}$$

where $[t_0, T]$ is the time interval of interest. The final weighting function $\phi(x(T), T)$ depends on the final state and final time and we want to make this function small. The weighting function $L(x, u, t)$ depends on the state and input at intermediate times in $[t_0, T]$. The performance index, which we will also refer to as the cost function, is chosen to make the plant (system) result in a minimum payoff.

The optimal control problem is to find the input $u^*(t)$ on the time interval $[t_0, T]$ that drives the plant (4.15) along a trajectory $x^*(t)$ such that the cost function (4.16) is minimized and

$$\psi(x(t_0), t_0) \;=\; 0 \tag{4.17}$$

for a given function $\psi \in \Re^p$.

To solve the continuous optimal control problem, we use Lagrange multipliers to adjoin the constraints (4.15) and (4.17) to the performance index (4.16). Since (4.15) holds at each $[t_0, T]$ we require an associated multiplier $\lambda(t) \in \Re^n$, which is a function of time. Since (4.17) holds only at one time, we require only a constant associated multiplier $v \in \Re^p$. The augmented performance index becomes

$$J' = \phi(x(T), T) + v^T \psi(x(T), T) + \int_{t_0}^{T} [L(x, u, t) + \lambda^T(t)(f(x, u, t) - x')]dt. \quad (4.18)$$

Let us now define the Hamiltonian function as

$$H(x, u, t) = L(x, u, t) + \lambda^T f(x, u, t), \quad (4.19)$$

then we can rewrite (4.18) as

$$J' = \phi(x(T), T) + v^T \psi(x(T), T) + \int_{t_0}^{T} [H(x, u, t) - \lambda^T x')]dt. \quad (4.20)$$

Using Leibniz's rule, the increment in J' as a function of increments in x, λ, v, u, and t is

$$\begin{aligned} dJ' = (\phi_x^T(T))dx(T) + (\phi_t(T)d(T) + (v^T \psi_x(T))dx(T) + \psi_t^T v dT + \psi^T(T)dv \quad (4.21) \\ +(H(T) - \lambda^T(T)x'(T))d(T) - (H(t_0) - \lambda^T x'(t_0))d(t_0) \\ + \int_{t_0}^{T} [H_x^T \delta x + H_u^T \delta u - \lambda^T \delta x' + (H_\lambda^T - x'^T)\delta\lambda]dt. \end{aligned}$$

To eliminate the variation in x', integrate by parts to see that

$$-\int_{t_0}^{T} \lambda^T \delta x' dt = -\lambda^T(T)\delta x(T) + \lambda^T(t_0)\delta x(t_0) + \int_{t_0}^{T} \lambda'^T \delta x \quad dt. \quad (4.22)$$

By substituting (4.22) into (4.21) we get terms at $t = T$ dependent on both $dx(t)$ and

$\delta x(T)$. We can express $\delta x(T)$ in terms of $dx(t)$ and dT using

$$dx(T) = \delta x(T) + x'(T)dT.$$

The result after substitution is

$$\begin{aligned} dJ' = (\phi_x^T(T))dx(T) + (\psi_x^T v)dx(T) - \lambda^T dx(T) + (\phi_t(T))d(T) \\ +(\psi_t^T(T)v)dT + H(T) + \psi_t^T dv - (H(t_0)d(t_0) + \lambda^T dx(t_0) \\ + \int_{t_0}^{T} [H_x^T \delta x + \lambda'^T \delta x(t_0 + H_u^T \delta u + (H_\lambda^T - x'^T)\delta\lambda]dt. \end{aligned} \tag{4.23}$$

According to the Lagrange Theory, the constrained minimum of J is attained at the unconstrained minimum of J'. This is achieved when $dJ = 0$ for all independent increments in its arguments. Setting the coefficients of the independent increments $dv, \delta x, \delta u$, and $\delta\lambda$ to zero yields necessary conditions for a minimum. In most texts for the applications highlighted, t_0 and $x(t_0)$ are both fixed and known so that dt_0 and $dx(t_0)$ are both zero. The two terms evaluated at $t = t_0$ in the above equation for dJ' are therefore automatically equal to zero.

In order to determine the optimal controller we need to determine the Hamiltonian, state equation, costate equation, stationarity condition, and boundary condition. The Hamiltonian is defined in (4.19). In general the state equation is

$$x' = \frac{\partial H}{\partial \lambda} = f, \quad t \geq t_0. \tag{4.24}$$

The costate equation develops backward in time, which in the continuous time case amounts to making the rate of change (i.e., λ') negative. The costate is also referred to as the adjoint to the state equation. The general form of the costate equation is

$$-\lambda' = \frac{\partial H}{\partial z} = \frac{\partial f^T}{\partial z}\lambda + \frac{\partial L}{\partial z}, \quad t \leq T. \tag{4.25}$$

The stationarity condition is written as

$$0 = \frac{\partial H}{\partial w} = \frac{\partial L}{\partial w} + \frac{\partial f^T}{\partial w}\lambda. \tag{4.26}$$

The boundary condition given $x(t_0)$ is given by the following equation

$$(\phi_x^T(T) + v^T\psi_x(T) - \lambda^T(T))dx(T) + (\phi_t(T) + \psi_t^T(T)v + H(T))dT = 0. \tag{4.27}$$

4.3.2 Our Particular Problem

System Model

We have the time interval $[0, \omega]$ and the following equations:

$$\begin{aligned} x' &= A(\theta)x + B(\theta)v + M\mu && (4.28a) \\ \hat{x}' &= A(\theta + \delta\theta)\hat{x} + B(\theta + \delta\theta)v + M\hat{\mu} && (4.28b) \\ 0 &= C(\theta)x + N\nu - C(\theta + \delta\theta)\hat{x} - N\hat{\nu}. && (4.28c) \end{aligned}$$

The state is represented by x and $\hat{x}$, v is the test signal, and $\mu, \hat{\mu}, \nu$, and $\hat{\nu}$ represent the additive uncertainty in the models. The only commonality between the two models is the detection signal, v, which is known and fixed. $x_0, \hat{x}_0$ are unknown. However, we will first assume x_0 and $\hat{x}_0$ are fixed and take the min over them. $\mu, \hat{\mu}, \nu$, and $\hat{\nu}$ are all controls (noises in this setup) and are unknown. To simplify notation we let $X_0 = X(\theta)$ and $X_1 = X(\theta + \delta\theta)$ for $X = A, B, C$, where subscript 0 corresponds to the normal model and subscript 1 corresponds to the faulty model. Therefore, we can rewrite (4.28) in the following manner:

$$x' = A_0x + B_0v + M\mu \tag{4.29a}$$

$$\hat{x}' = A_1\hat{x} + B_1v + M\hat{\mu} \tag{4.29b}$$

$$0 = C_0x + N\nu - C_1\hat{x} - N\hat{\nu}. \tag{4.29c}$$

Note in our problem we actually have multiple controls $\mu, \hat{\mu}, \nu$ and $\hat{\nu}$. However, we will later eliminate ν and $\hat{\nu}$ in the cost function so we will be left with only two vector controls μ and $\hat{\mu}$.

Cost Function

Our performance index for the optimal control problem is represented using the following equation:

$$J(u) = \phi(x(t_0), t_0) + \int_{t_0}^{T} L(x, u, t)dt. \tag{4.30}$$

In most texts $\phi(x(T), T)$ is used, but in our application we have $\phi(x(t_0), t_0)$. Before any simplification our cost function is

$$J = \min\{x(0)^Tx(0) + \hat{x}(0)^T\hat{x}(0) + \int_0^{\omega} \nu^T\nu + \hat{\nu}^T\hat{\nu} + \mu^T\mu + \hat{\mu}^T\hat{\mu}dt\}. \tag{4.31}$$

In order to reduce the number of unknowns in the problem, we start with (4.29c) and solve for $\hat{\nu}$ in order to eliminate $\hat{\nu}$ from J by replacing it with its substitution, which is in terms of $x, \hat{x}$, and ν. Start by adding $N\hat{\nu}$ to both sides of (4.29c) to get

$$N\hat{\nu} = C_0x + N\nu - C_1\hat{x}. \tag{4.32}$$

Now we multiply each term by N^{-1} and use substitutions $\tilde{C}_i = N^{-1}C_i$ and $\tilde{C}_i^{\,T} = C_i^T N^{-T}$, $i = 0, 1$. Then (4.32) reduces to

$$\hat{\nu} \;=\; \nu + \tilde{C}_0 x - \tilde{C}_1 \hat{x} \tag{4.33}$$

since $N^{-1}N$ is just the identity matrix and $(N^{-1})^T = N^{-T}$. However, we can rewrite (4.33) as

$$\hat{\nu} \;=\; \nu + \tilde{C} z, \tag{4.34}$$

with

$$\tilde{C}^T = \begin{pmatrix} \hat{C}_0^{\,T} \\ -\hat{C}_1^{\,T} \end{pmatrix}, \quad z = \begin{pmatrix} x \\ \hat{x} \end{pmatrix}.$$

Then (4.34) is substituted into (4.31) to get (4.35). Once we simplify and replace $\hat{\nu}$ with its substitution our cost function can be written this way:

$$J \;=\; \min z(0)^T z(0) + \int_0^{\omega} (w^T w + \nu^T \nu + (\nu + \tilde{C}z)^T(\nu + \tilde{C}z))dt, \tag{4.35}$$

where

$$w = \begin{pmatrix} \mu \\ \hat{\mu} \end{pmatrix}.$$

Equation (4.29c) has now been eliminated. The constraints are just (4.28a), (4.28b). Note that ν does not appear in the constraints. Thus we can minimize the ν term in J directly. Differentiating (4.35) with respect to ν we get

$$\nu \;=\; -\frac{1}{2}\tilde{C}z. \tag{4.36}$$

Now, the cost function becomes the minimum of

$$\begin{aligned} J &= z(0)^T z(0) + \int_0^\omega (w^T w + \nu^T \nu + (\nu + \tilde{C}z)^T (\nu + \tilde{C}z))dt \\ &= z(0)^T z(0) + \int_0^\omega (w^T w + \frac{1}{4} z^T \tilde{C}^T \tilde{C} z + \frac{1}{4} z^T \tilde{C}^T \tilde{C} z)dt \\ &= z(0)^T z(0) + \int_0^\omega (w^T w + \frac{1}{2} z^T \tilde{C}^T \tilde{C} z)dt. \end{aligned}$$

Now, we have our cost function in terms of only two unknowns μ and $\hat{\mu}$ and we represent them using vector w. The combined system can be written in this way:

$$J = \min z(0)^T z(0) + \int_0^\omega (w^T w + \frac{1}{2} z^T Q z)dt, \tag{4.38}$$

where

$$Q = \tilde{C}^T \tilde{C} = \begin{pmatrix} \hat{C_0}^T \hat{C}_0 & -\hat{C_0}^T \hat{C}_1 \\ -\hat{C_1}^T \hat{C}_0 & \hat{C_1}^T \hat{C}_1 \end{pmatrix}$$

and Q is a positive semi-definite symmetric matrix.

Our problem no longer has a state constraint.

4.3.3 Optimal Controller

Hamiltonian

The general form of the Hamiltonian for the combined system is

$$H(z, w, t) = L(z, w, t) + \lambda^T f(z, w, t). \tag{4.39}$$

Before eliminating the control variable ν and simplifying the Hamiltonian expression we want to solve for ν using the stationarity condition, $H_\nu = 0$. Before any simplification,

$$\begin{aligned}H = \mu^T\mu + \hat{\mu}^T\hat{\mu} + 2\nu^T\nu + x^T\hat{C_0}^T\hat{C}_0 x - \hat{x}^T\hat{C_1}^T\hat{C}_0 x + \nu^T\hat{C}_0 x - x^T\hat{C_0}^T\hat{C}_1\hat{x} + \hat{x}^T\hat{C_1}^T\hat{C}_1\hat{x}\\ -\nu^T\hat{C}_1\hat{x} + x^T\hat{C_0}^T\nu - \hat{x}^T\hat{C_1}^T\nu + \lambda_0^T(A_0 x + B_0 v + M\mu) + \lambda_1^T(A_1\hat{x} + B_1 v + M\hat{\mu}).\end{aligned}$$

Then using our stationarity condition

$$0 = H_\nu = 4\nu^T + 2x^T\hat{C_0}^T - 2\hat{x}^T\hat{C_1}^T, \tag{4.40}$$

which implies that

$$0 \;=\; 4\nu^T + 2z^T\tilde{C}^T. \tag{4.41}$$

When we solve for ν we get

$$\nu \;=\; -\frac{1}{2}\tilde{C}z. \tag{4.42}$$

For the combined system we assume

$$A = \begin{pmatrix} A_0 & 0 \\ 0 & A_1 \end{pmatrix}, \quad z = \begin{pmatrix} x \\ \hat{x} \end{pmatrix}, \quad \hat{M} = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix}, \quad w = \begin{pmatrix} \mu \\ \hat{\mu} \end{pmatrix}, \quad B = \begin{pmatrix} B_0 \\ B_1 \end{pmatrix}.$$

Once we combine the normal and faulty systems into one big system and simplify, we get this Hamiltonian equation:

$$H \;=\; w^T w + \frac{1}{2}z^T Q z + \lambda^T(Az + \hat{M}w + Bv). \tag{4.43}$$

For linear time invariant systems and cost functions, the Hamiltonian is constant. Therefore, $H' = 0$. The general form for the time derivative of the Hamiltonian is

$$H' = H_t + H_z^T z' + H_w^T w' + \lambda'^T f = H_t + H_w^T w' + (H_z + \lambda')^T f. \tag{4.44}$$

Note that if $w(t)$ is an optimal control, then $H' = H_t$. Since $H_t = 0$ our expression becomes:

$$H' = H_w^T w' + H_z^T f + \lambda'^T f, \tag{4.45}$$

which gives

$$H' = 2w^T w' + \lambda^T \hat{M} w' + z^T Q(Az + \hat{M}w + Bv) + \lambda'^T (Az + \hat{M}w + Bv). \tag{4.46}$$

State Equation

Our combined system model can be written in the following manner:

$$z' = Az + \hat{M}w + Bv. \tag{4.47}$$

Costate Equation

$$\lambda' = -Q^T z - A^T \lambda. \tag{4.48}$$

Stationarity Conditions

For our problem, v is fixed and in our reduced problem our control variable is vector w. Thus, we write the stationarity condition in the following way:

$$0 = H_w = 2w^T + \lambda^T \hat{M}. \tag{4.49}$$

This implies that

$$w \quad = \quad -\frac{1}{2}\hat{M}^T \lambda. \tag{4.50}$$

Boundary Condition

Given (4.15)-(4.30) we can derive our boundary conditions for this problem. We start with the augmented performance index:

$$J' = \phi(z(t_0), t_0) + \phi(z(T), T) + \int_{t_0}^{T} [L(z, u, t) + \lambda^T(t)(f(z, u, t) - z')]dt. \tag{4.51}$$

Then if we define the Hamiltonian as in (4.39) and J' becomes

$$J' = \phi(z(t_0, t_0) + \phi(z(T), T) + \int_{t_0}^{T} [H(z, u, t) - \lambda^T z')]dt. \tag{4.52}$$

Using Leibniz's rule, the increment in J' as a function of increments in z, λ, v, u, and t is

$$\begin{aligned} dJ' = (\phi_z^T(t_0))dz(t_0) + (\phi_t(t_0)d(t_0) + (\phi_z(T))dz(T) + (\phi_t(T))d(T) \\ +(H(T) - \lambda^T(T)z'(T))d(T) - (H(t_0) - \lambda^T z'(t_0))d(t_0) \\ + \int_{t_0}^{T} [H_z^T \delta z + H_u^T \delta u - \lambda^T \delta z' + (H_\lambda^T - z'^T)\delta\lambda] dt. \end{aligned} \tag{4.53}$$

To eliminate z', integrate by parts to see that

$$-\int_{t_0}^{T} \lambda^T \delta z' dt = -\lambda^T(T)\delta z(T) + \lambda^T(t_0)\delta z(t_0) + \int_{t_0}^{T} \lambda'^T \delta z dt. \tag{4.54}$$

Then,

$$\begin{aligned} dJ' = (\phi_z^T(t_0) + \lambda^T(t_0))dz(t_0) + (\phi_t(t_0) - H(t_0))dt_0 + (\phi_z^T(T) - \lambda^T(T))dz(T) \\ +(\phi_t(T) + H(T))dT + \int_{t_0}^{T} [(H_z^T + \lambda'^T)\delta z + H_u^T \delta u + (H_\lambda^T - z'^T)\delta\lambda] dt. \end{aligned} \tag{4.55}$$

Based on this derivation we get the initial and terminal boundary conditions for the LQR problem. By Lagrange theory, the constrained minimum of J is attained at the unconstrained minimum of J'. This is achieved when $dJ' = 0$ for all independent increments in its arguments. Setting $dv = 0, \delta z = 0, \delta u = 0$, and $\delta\lambda = 0$ yields the necessary conditions for the minimum as shown above. Since t_0 and T are fixed, $d(t_0) = 0$ and $d(T) = 0$. Therefore, the terms with those values in it drop out. However, $dz(t_0)$ and $dz(T)$ are free. There is no ψ in this problem so those values are zero. Further, $\phi(T) = 0$, but $\phi(t_0)$ is not zero. So once we get rid of the terms that drop out, we can rewrite dJ' as

$$dJ' \;=\; (\phi_z^T(t_0) + \lambda^T(t_0))dz(t_0) - \lambda^T(T)dz(T). \tag{4.56}$$

Recall our time interval is $[0, \omega]$. We are also given $z(0)$. We find the initial condition by evaluating

$$(\phi_z^T(0) + \lambda^T(0))dz(0) = 0 \tag{4.57}$$

and $dz(0) = 0$ since $z(0)$ is fixed. Therefore, $\phi_z^T(0) + \lambda^T(0)$ is free.

However, if $z(0)$ is not given, then

$$(\phi_z^T(0) + \lambda^T(0)) = 0. \tag{4.58}$$

By doing some algebra and taking the transpose of every term

$$\lambda(0) = -\phi_z(0), \tag{4.59}$$

but based on (4.50), we know $\phi_z(0) = 2z(0)$. Therefore,

$$\lambda(0) = -2z(0). \tag{4.60}$$

The term that is left gives us the terminal condition. To find the boundary condition at the final time, ω, we use

$$-\lambda^T(\omega)dz(\omega) = 0. \tag{4.61}$$

$dz(\omega)$ is free and if we transpose the remaining term we are left with

$$\lambda(\omega) = 0. \tag{4.62}$$

Our boundary conditions when $z(0)$ is not fixed are:

$$\begin{aligned} \lambda(0) &= -2z(0) && (4.63) \\ \lambda(\omega) &= 0, && (4.64) \end{aligned}$$

where $\lambda(\omega)$ has the same dimension as z.

4.4 Solving the Inner Minimization Problem

Based on the solution to the optimal controller found in Section 4.3 we get the set of vectors and matrices found below, which will be used to solve the inner optimization problem. Note that we want to find the worst fault of the LQR problem for one big system, which includes the set of equations for the normal and faulty models. We need to minimize the cost function J in Equation (4.65).

Let

$$z = \begin{pmatrix} x \\ \hat{x} \end{pmatrix}, \quad w = \begin{pmatrix} \mu \\ \hat{\mu} \end{pmatrix}, \quad Q = \begin{pmatrix} \hat{C_0}^T \hat{C_0} & -\hat{C_0}^T \hat{C_1} \\ -\hat{C_1}^T \hat{C_0} & \hat{C_1}^T \hat{C_1} \end{pmatrix},$$

$$A = \begin{pmatrix} A_0 & 0 \\ 0 & A_1 \end{pmatrix}, \quad \hat{M} = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix}, \quad B = \begin{pmatrix} B_0 \\ B_1 \end{pmatrix}.$$

There are two approaches for solving the inner minimum problem. In Sections 4.4.1 and 4.4.2 we outline each approach.

4.4.1 Approach I

Using this approach we temporarily fix $z(0)$ and that gives us that $dz(0) = 0$. We use LQR theory to solve the minimization problem. Then, we perform another layer of minimization with respect to $z(0)$.

Inner Optimization Problem - Original Problem in the Lewis Form

Based on the work presented in Section 4.3, we begin with the following cost function and state equation which were found by determining the solution to our optimal control problem:

$$J = \min z(0)^T z(0) + \int_0^{\omega} (w^T w + \frac{1}{2} z^T Q z) dt \tag{4.65}$$

$$z' = Az + Bv + \hat{M} w \tag{4.66}$$

The cost functional in equation (4.65) is the minimum over $z(0), w$ such that (4.66) holds.

Solution

In order to determine the solution for the optimal control w, we want Equation (4.65) in the form of the continuous linear quadratic tracker problem. The control input is an affine state feedback, i.e., it consists of a linear state feedback plus an additional term. Let $r(t)$ be the reference track. The additional term depends on the output $u(t)$ of the costate of the closed-loop plant when driven by $r(t)$. Note that without the fixed function Bv in our system, our problem is like the regular LQR problem, except $\hat{M}$ is the coefficient matrix for our control instead of B. Assume

$$r' = Ar + Bv, \quad r(0) = 0. \tag{4.67}$$

Let

$$\bar{z} = z - r. \tag{4.68}$$

Then,

$$\begin{aligned}\bar{z}' &= z' - r' &(4.69)\\ &= Az + Bv + \hat{M}w - Ar - Bv\\ &= Az - Ar + \hat{M}w\\ &= A\bar{z} + \hat{M}w.\end{aligned}$$

Now (4.65) can be rewritten as

$$J = \min(\bar{z}(0) + r(0))^T(\bar{z}(0) + r(0)) + \int_0^\omega \frac{1}{2}(\bar{z} + r)^T Q(\bar{z} + r) + w^T w dt. \quad (4.70)$$

This optimization is done in two stages: inner minimization with fixed $\bar{z}(0)$ and outer optimization over $\bar{z}(0)$. Note that $z(0)$ is now fixed and $z(\omega)$ is free. We want $z(\omega)$ close to zero at the final time since $r(\omega) = 0$.

Following Table 4.1-1 in [26], $P \geq 0, Q \geq 0, R > 0$ are all symmetric matrices. Also, K is the Kalman gain matrix, (4.71b) is the Ricatti equation, and w is our control. The optimal affine control in terms of time t is given by:

$$\begin{aligned}K(t) &= R^{-1}\hat{M}^T\Pi(t) &(4.71a)\\ \Pi' &= -A^T\Pi - \Pi A + \Pi\hat{M}R^{-1}\hat{M}^T\Pi - C^TQC, \quad \Pi(0) = C^TPC &(4.71b)\\ u' &= (-A + \hat{M}K)^T u - C^TQr, \quad u(0) = C^TPr(0) &(4.71c)\\ w &= -K\bar{z} + R^{-1}\hat{M}^T u. &(4.71d)\end{aligned}$$

Note that here control is actually the smallest noise given the same output.

Change of Variables

In the continuous time case, the Ricatti equation must be integrated backward. Most integration routines run forward in time so we convert the Ricatti equation into an equation that is integrated forward. To put this into a more standard form we perform a change of variables and let $\tau = \omega - t$. Then since ω is a fixed value,

$$d\tau = -dt.$$

Now we assume

$$r' = -Ar - Bv, \quad r(\omega) = 0, \tag{4.72}$$

where we know what we want our reference to be at the final time ω.

Let

$$\bar{z} = z - r. \tag{4.73}$$

Then,

$$\begin{aligned} \bar{z}' &= z' - r' \\ &= -Az - Bv - \hat{M}w + Ar + Bv \\ &= -Az + Ar - \hat{M}w \\ &= -A\bar{z} - \hat{M}w. \end{aligned} \tag{4.74}$$

The cost function in terms of the final time

$$J = \min(\bar{z}(\omega) + r(\omega))^T(\bar{z}(\omega) + r(\omega)) + \int_0^\omega \frac{1}{2}(\bar{z} + r)^T Q(\bar{z} + r) + w^T w d\tau. \tag{4.75}$$

Since $r(\omega) = 0$ we will replace (4.75) by

$$J = \min \bar{z}(\omega)^T \bar{z}(\omega) + \int_0^\omega \frac{1}{2}(\bar{z} + r)^T Q(\bar{z} + r) + w^T w d\tau. \tag{4.76}$$

For the remainder of this section z will be in the new variable τ as will all differential equations and derivatives. Note that $z(0)$ and $z(\omega)$ are both free. We shall fix $z(0)$ and then reformulate to get a standard problem. Then given the solution of the standard problem we will minimize over $z(0)$.

Solution to Optimal Control

Again using Table 4.1-1 in [26] and simplifying based on our system we follow the form in [26] $P = 2I, R = 2I$, and $C = I$, where I is an identity matrix of appropriate dimensions. The optimal affine control can be found using the following set of equations:

$$K(\tau) = -\frac{1}{2}\hat{M}^T \Pi(\tau) \tag{4.77a}$$

$$\Pi' = A^T \Pi + \Pi A + \frac{1}{2}\Pi \hat{M}\hat{M}^T \Pi - Q, \quad \Pi(\omega) = 2I \tag{4.77b}$$

$$u' = (A - \hat{M}K)^T u - Qr, \quad u(\omega) = 0 \tag{4.77c}$$

$$w = -K\bar{z} - \frac{1}{2}\hat{M}^T u. \tag{4.77d}$$

Here we take the normal Ricatti equation and integrate it forward in time from $t = 0$ without the minus sign on the left-hand side, then reverse the resulting solution to shift it to $t = \omega$. Further details about using this setup can be found on pages 174-175 of [26]. This is equivalent to keeping the Ricatti equation as $-\Pi'$ and then taking the integral from ω to 0, where now 0 is our upper bound.

The closed loop plant becomes:

$$\bar{z}' = (A - \hat{M}K)\bar{z} + \hat{M}\hat{M}^T u. \tag{4.78}$$

The optimal cost on $[\tau, \omega]$ for any τ using this control is

$$J(\tau) = \frac{1}{2}\bar{z}(\tau)^T \Pi(\tau)\bar{z}(\tau) - \bar{z}(\tau)^T u(\tau) + f(\tau), \tag{4.79}$$

where the new auxiliary function $f(\tau)$ satisfies

$$f' = \frac{1}{4}u\hat{M}\hat{M}^T u - \frac{1}{2}r^T Q r, \quad \tau \leq \omega \tag{4.80}$$

with

$$f(\omega) = r(\omega)^T r(\omega) = 0$$

since our reference track at the final time is $r(\omega) = 0$. The optimal control w is determined by solving (4.77b) for Π. Since $\bar{z}$ is not required to find Π, this calculation can be done off-line. The Kalman gain matrix K can also be computed and stored. Note that $K(t)$ is completely specified once the system, cost function, and terminal time are specified. During the control run the optimal control w^* is found using (4.77d) applied to the plant which gives (4.78). However, since our control is the additive uncertainty or noise in our model we only need to minimize the cost function. We just make sure that the noise is within the bounds set.

The optimal cost to go and the value of the inner min will be

$$J(0) = \frac{1}{2}\bar{z}(0)^T \Pi(0)\bar{z}(0) - \bar{z}(0)^T u(0) + f(0). \tag{4.81}$$

Now we minimize over $\bar{z}(0)$. Taking the derivative of $J(0)$ with respect to $\bar{z}(0)$ we get

$$J_{\bar{z}(0)} = \Pi(0)\bar{z}(0) - u(0) \tag{4.82}$$

and setting $J_{\bar{z}(0)} = 0$ gives

$$\bar{z}(0) = \Pi^{-1}(0)u(0). \tag{4.83}$$

We can simplify by substituting (4.83) into (4.81). The cost becomes

$$J(0) = -\frac{1}{2}u(0)^T\Pi(0)^{-1}u(0) + f(0). \tag{4.84}$$

It is important to note that all we need from this inner minimization is the inner cost. As shown in this section that can be accomplished by a set of numerical integrations. Depending on the length of the time interval and number of grid points we have the option of either integrating a second time or saving the previous integration if we need a function a second time.

Algorithm for the Inner Minimization Using Approach I

The previous formulations and calculations were done to determine a solution for the optimal test signal v that works. To actually carry out the evaluation the following must be done. Note that we start out knowing $r(\omega), f(\omega), \Pi(\omega), u(\omega)$.

1. Integrate (4.77b), (4.77c), (4.72) in backward time to get $\Pi(0), u(0), r(0)$.
2. The feedback K (or Π) is saved from this iteration.
3. Equation (4.83) now gives us $\bar{z}(0)$. In order to evaluate (4.84) we need $f(0)$.

4. Given $\bar{z}(0)$ and $r(0)$ we can integrate (4.69), (4.77b), and (4.77c) in forward time to get $\bar{z}(\omega)$.

5. We can now integrate (4.80) in backward time to get $f(0)$.

Note that since we are interested in the value of the minimum, the results from the integrations do not need to be saved unless it is advantageous in some way. Also, the length of the integration interval is the test period which usually will not be very long. The accuracy required of the numerical integration is related to the size of the incipient faults and the conditioning of the differential equations of the system tested. Accordingly, we use a fourth order integrator to solve this problem numerically. If the systems are not stiff, an explicit method will work faster. For integrations involving v, we have that v is piecewise constant. To maintain accuracy our first choice is a fourth order Runge-Kutta method, where the computational grid is chosen so that the points t_i where v changes are a part of the grid. Since the Runge-Kutta methods are one step methods, they will not lose any accuracy due to the loss of smoothness in v.

The integrations will be done many times since they are in the inner minimization. For many problems they can be sped up by either using a fixed grid, or computing the grid on the first integration, and then for subsequent iterations reusing the grid found for the first $\delta\theta$ value.

The computational effort can also be sped up by using an educated guess for the first value of v. For example, one could pick a value of $\delta\theta$ in the interior of Ω, use the standard two model algorithm to find a good proper test signal, and then use this signal to begin the optimization. When computing this initial guess it is helpful to take $\|\delta\theta\|$ as small as possible since this will produce a larger and more realistic initial guess for v.

Depending on the incipient faults of interest there may be additional ways to speed up the algorithm. For example, sensor faults would often be modeled by matrix C depending on θ.

4.4.2 Approach II

Ricatti solutions are popular since they restrict the size of the problem that is being dealt with at each time step and because they provide the optimal solution in a feedback form. However, in our setting when conducting the inner min we are only interested in the value of the optimal cost. In the inner min, the control is the worse case noise and we do not need to compute it. Also, the testing period may be short. Thus, in some cases it may be better to directly solve for the minimum value from the necessary conditions.

1. *Transformed Problem:* The boundary value problem (bvp) and necessary conditions follow. Based on the work done using Approach I for solving the inner min, we have

$$\bar{z}' = -A\bar{z} - \hat{M}w \tag{4.85a}$$
$$r' = -Ar - Bv, \quad r(\omega) = 0. \tag{4.85b}$$

   We also have our Hamiltonian

$$H = \frac{1}{2}(\bar{z}+r)^T Q(\bar{z}+r) + w^T w - \lambda^T(-A\bar{z} - \hat{M}w). \tag{4.86}$$

   Thus,

$$\lambda' = A^T\lambda - Q\bar{z} - Qr, \tag{4.87a}$$
$$w = \frac{1}{2}\hat{M}^T\lambda. \tag{4.87b}$$

   So we can rewrite the state equation as

$$\bar{z}' = -A\bar{z} - \frac{1}{2}\hat{M}\hat{M}^T\lambda \tag{4.88}$$

and the cost function can be written as

$$J = \bar{z}(\omega)^T \bar{z}(\omega) + \int_0^\omega \frac{1}{2}(\bar{z}+r)^T Q(\bar{z}+r) + \frac{1}{4}\hat{M}\hat{M}^T \lambda d\tau \tag{4.89}$$

The following is the algorithm for the inner minimization using Approach II. We solve the boundary value problem

$$\begin{aligned}
\bar{z}' &= -A\bar{z} - \frac{1}{2}\hat{M}\hat{M}^T\lambda, && \text{(4.90a)}\\
\lambda' &= A^T\lambda - Q\bar{z} - Qr, \quad \lambda(\omega) = 2\bar{z}(\omega) && \text{(4.90b)}\\
r' &= -Ar - Bv, \quad r(\omega) = 0 && \text{(4.90c)}\\
q' &= \frac{1}{2}(\bar{z}+r)^T Q(\bar{z}+r) + \frac{1}{4}\lambda^T\hat{M}\hat{M}^T\lambda, \quad q(0) = 0 && \text{(4.90d)}
\end{aligned}$$

with a boundary value solver. Given the solution of (4.90), the value of the inner minimum for fixed $z(0)$ is

$$q(\omega) + \bar{z}(\omega)^T\bar{z}(\omega). \tag{4.91}$$

Suppose that the dimension of $\bar{z}$ is n. Then, just one integration of (4.77b) requires integrating a system with $\frac{1}{2}(n^2+n)$ unknowns. On the other hand, (4.90) involves $3n+1$ dimensional differential equation. Thus, using (4.90) could be competitive under some conditions.

2. *Untransformed Problem:* If it is decided to use a boundary value problem solver then we do not need several of the earlier transformations used to get a tracking problem in standard form. Instead we can start with the optimal control problem (4.35) subject to (4.66). Since ν appears in the cost function, but not in the constraint we can first minimize over ν and get again that $\nu = \frac{1}{2}\tilde{C}z$. Thus

we have to minimize

$$J = z(0)^T z(0) + \int_0^\omega w^T w + \frac{1}{2} z^T \tilde{C}^T \tilde{C} z d\tau \tag{4.92}$$

subject to

$$z' = Az + Bv + \hat{M}w. \tag{4.93}$$

The Hamiltonian is given by

$$H = w^T w + \frac{1}{2} z^T \tilde{C}^T \tilde{C} + \lambda^T (Az + Bv + \hat{M}w) \tag{4.94}$$

so that the necessary conditions include

$$\begin{aligned} z' &= -Az + Bv + \hat{M}w && \text{(4.95a)} \\ \lambda' &= -A^T \lambda - \tilde{C}^T \tilde{C} z && \text{(4.95b)} \\ 0 &= 2w + \hat{M}^T \lambda. && \text{(4.95c)} \end{aligned}$$

Using (4.95c) to eliminate w we get that the boundary value solver call will solve

$$\begin{aligned} z' &= Az + Bv - \frac{1}{2} \hat{M} \hat{M}^T \lambda && \text{(4.96a)} \\ \lambda' &= -A^T \lambda - \tilde{C}^T \tilde{C} z && \text{(4.96b)} \\ u' &= \frac{1}{4} \lambda^T \hat{M} \hat{M}^T \lambda + \frac{1}{2} z^T \tilde{C}^T \tilde{C} z && \text{(4.96c)} \\ \lambda(0) &= -\frac{1}{2} z(0) && \text{(4.96d)} \\ \lambda(\omega) &= 0 && \text{(4.96e)} \\ u(0) &= 0. && \text{(4.96f)} \end{aligned}$$

The value of the inner minimum is then

$$u(\omega) + z(0)^T z(0). \tag{4.97}$$

4.5 Outer Optimization

The outer optimization is done over the test signal. We begin with a test signal v that is infinite dimensional so we need to reduce the problem to a finite dimensional problem. Here we discuss how to compute v and a method used to develop a finite problem.

4.5.1 Computation of the Test Signal

We now carefully examine how to evaluate the smallest noise $S(v, \delta\theta)$. The outline of our general algorithm follows:

- For fixed $\delta\theta, v$ with $\|v\| = 1$ we compute the minimum of S where S is given by (4.11). That is, we compute the smallest amount of noise consistent with getting the same output from both models for a given value of $\delta\theta$. This is the first min. Note that we only need the value and not the noise that produces that value. We call this minimal value $S(v, \delta\theta)$.

- Then, we minimize $S(v, \delta\theta)$ over $\delta\theta \in \Omega$. For a given v we derive the smallest noise S that results in equal outputs for all $\delta\theta$. The value that results from this calculation is called $S(v)$ or the second minimum.

- The larger the value for $S(v)$, the better the resulting test signal. Accordingly, we perform the outer maximum over $\|v\| = 1$ of $S(\delta\theta)$ to get a value S_* and a test signal $\bar{v}$ satisfying the condition $\|\bar{v}\| = 1$.

- If $S_* > 1$, then $\bar{v}$ is proper but can be taken smaller. If $S_* < 1$, then $\bar{v}$ is not proper and must be increased. Finally, we determine an optimal proper incipient test signal $v^* = \frac{\bar{v}}{\sqrt{S_*}}$.

Ω is parameterized by a finite dimensional vector $\delta\theta$. Thus, in principle this minimum can be carried out by standard routines in Matlab such as fmincon as long as there is a subroutine to evaluate the smallest noise $S(v, \delta\theta)$.

The outer optimization is done over v. In general, v is infinite dimensional. However, the problem can be reduced to a finite dimensional problem using finite dimensional approximations, which often produce test signals that are both easier to implement [12] and sufficiently optimal. Two of the more popular finite dimensional approximations are piecewise constant (or piecewise spline) test signals or truncated Fourier series. However, piecewise constant (or piecewise linear) signals are often used in practice since they are easier to implement. If we take v to be from a finite dimensional signal space, then this outer maximum is also over a finite dimensional compact set. There are a many ways to get a finite dimensional family of v, which is best depends on the problem. This idea is discussed further in Section 4.5.2.

4.5.2 Piecewise Constant Controls

Often it is most practical to apply piecewise constant or piecewise linear inputs. We consider piecewise constant test signals. Assume v is piecewise constant, then there are

- A finite number of times $0 = t_0 < t_1 < \ldots < t_k = \omega$,
- A finite number of values v_i such that $v(t) = v_i$ for $t \in [t_i, t_{i+1})$.

In the first variation, the times $t_0, \ldots, t_k$ are fixed and the values $v_0, \ldots, v_{k-1}$ are the parameterization of the test signal we optimize over. If the v_i are m-dimensional, then there are mk parameters. Often we can take k small, like $k = 3$ or $k = 4$. In the second variation k is fixed, but the times $t_1, \ldots, t_{k-1}$ are not fixed. The values

$v_0, \ldots, v_{k-1}$ and the $t_1, \ldots, t_{k-1}$ are the parameterization of the test signal we optimize over. If the v_i are m-dimensional then there are $mk + k - 1$ parameters. Here we expect to be able to take k a little smaller than in variation one.

In [12] an algorithm is presented to compute optimal piecewise constant test signals. This paper focused on the case where the test signal changes a small number of times. Examples were shown to illustrate computational differences between different types of piecewise constant signals.

In [12] Choe et. al. develop a minimal piecewise constant test signal of the original continuous time fault detection problem. With continual monitoring using a test signal v, a fault that causes an abrupt change in the system is detected. The times where the v changes value is called the v-grid. There are two steps in this process:

- Develop an algorithm for computing minimal piecewise constant detection signals for a fixed v-grid;
- Optimize over the v-grid of the signal.

Note that the v-grid optimization is a nonlinear inequality constrained optimization problem.

[12] highlights the case where the detection signal has a small number of constant pieces, like one or two. As long as the uncertainty bounds and model assumptions hold, the computed test signal is designed to guarantee the correct decision. There is no particular form to the uncertainty.

4.5.3 Optimizing Over the Parameters

Provided the set Ω has a reasonable form, the next level of minimization is straightforward. However, we must be careful when we do the outer optimization since it is over $\|v\| = 1$. The approach taken depends on what optimizers we have available. We show the case where $t_{i+1} - t_i = \tau$ a constant and we have times $t_0, \ldots, t_K$. We assume $\gamma = \sqrt{\tau}^{-1}$.

One parameterization is

$$\begin{aligned} \|v_0\| &\leq \gamma && \text{(4.98a)} \\ \|v_1\| &\leq \gamma\sqrt{1-\|v_0\|^2} && \text{(4.98b)} \\ \|v_2\| &\leq \gamma\sqrt{1-\|v_0\|^2-\|v_1\|^2} && \text{(4.98c)} \\ &\vdots && \text{(4.98d)} \end{aligned}$$

Now for each individual v_i we can use n-dimensional spherical coordinates. This results in K identical sets of box constraints and the radial coordinate bounds:

$$\begin{aligned} \rho_0 &\leq \gamma && \text{(4.99a)} \\ \rho_1 &\leq \gamma\sqrt{1-\rho_0^2} && \text{(4.99b)} \\ \rho_2 &\leq \gamma\sqrt{1-\rho_0^2-\rho_1^2} && \text{(4.99c)} \\ &\vdots && \text{(4.99d)} \\ \rho_k &\leq \gamma\sqrt{a-\sum_{i=0}^{K-1}\|\rho_i\|^2}. && \text{(4.99e)} \end{aligned}$$

It is important to note that for a given v that $\frac{v}{\sqrt{S(v)}}$ is a proper test signal. Thus during the optimization over d we can stop at any time and have a test signal that is proper for all incipient faults with $\delta\theta \in \Omega$. Attaining a minimum proper test signal will usually not be necessary. A sufficiently small detection signal will work.

If $S(v) > 0$, then some multiple of v will be proper. This means that during the outer optimization at each iteration we can easily get a proper test signal which is not optimal. Accordingly, early termination of the algorithm results in a test signal that is suboptimal, but could still be quite good. In practice this feature should be utilized in the code to compute the test signal.

4.6 Examples

Example 7. *To illustrate how we can implement the procedure outlined in Section 4.5.1 we consider the following academic example on the interval* $[0, L]$*. We suppose* $L = 4$ *for the following system*

$$x' = \begin{pmatrix} -\theta_1 & \theta_2 \\ 1 & -\theta_1 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} v + \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \tag{4.100a}$$

$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} x + \nu_1. \tag{4.100b}$$

A change in θ_1 *may be thought of as a change in the internal damping of friction of the system while a change in* θ_2 *alters a frequency of the damped free response. We suppose that our test signal generator produces sinusoids of predetermined frequency and we seek a test signal of the following form*

$$v = c_1 g \sin(\frac{\pi t}{L}) + c_2 g \sin(\frac{2\pi t}{L}) + \ldots + c_n g \sin(\frac{n\pi t}{L}), \tag{4.101}$$

where $g = \sqrt{\frac{2}{L}}$ *and* n *is any positive integer. The* v *are thus chosen in a finite dimensional subspace.*

We compute optimal test signals for $n = 1, 2, 3, 5$*. We can think of* v *as using the first* n *terms of a Fourier sine series to generate a finite dimensional space of test signals. Then* $v(t)$ *has* L^2 *norm of one precisely when* $c_1^2 + c_2^2 + \ldots + c_n^2 = 1$*.*

We parameterize c using the same box constraints we had over w in the discrete time case. $\|c\| = 1$ *and the parameterization is*

$$
\begin{aligned}
c_1 &= \cos(d_1) && (4.102)\\
c_2 &= \sin(d_1)\cos(d_2) && (4.103)\\
c_3 &= \sin(d_1)\sin(d_2)\cos(d_3) && (4.104)\\
&\vdots\vdots && (4.105)\\
c_{n-1} &= \sin(d_1)\ldots\sin(d_{n-2})\cos(d_{n-1}) && (4.106)\\
c_n &= \sin(d_1)\ldots\sin(d_{n-1}) && (4.107)
\end{aligned}
$$

where $0 \le d_1 \le \pi$ *and* $0 \le d_i \le 2\pi$ *for* $i \in [2, n-1]$. *We refer to the* $c_i^\prime s$ *as the coefficients of* v. *Note that the test signal is a function of* c *and* t, *i.e.,* $v(c,t)$. *The* $n = 1$ *case of Equation (4.101) is also of interest. In that case there is no optimization over* d, *just a rescaling is needed to make* v *proper.*

The normal parameter values are $\theta_1 = 0.1, \theta_2 = 1.0$, *which gives eigenvalues of* $-\theta_1 \pm \sqrt{\theta_2}$. *The fault condition is given by* $\delta\theta_1 \in [1,2]$ *and* $\delta\theta_2 \in [2,4]$.

To solve this problem we use the second algorithm outlined in Section 4.4.2. The boundary value problem (4.96) is solved using the Matlab function called bvp4c and it gives a solution to the innermost minimum. Then we optimize over the perturbations and over the test signals using a Matlab command called fmincon. This gives the *optimal test signal v^*. This problem is well behaved, but if the dynamics are stiff then extra care is needed when integrating the boundary value problem.*

In fact, initially we started with a different matrix A, but a problem occurred when the optimizer was applied to the system because the solution was getting too large too fast. This error was due to the eigenvalues of matrix A, which resulted in an ill-conditioned boundary value problem and the solver would not produce any solutions. In order to resolve this problem we changed A so that it would have smaller eigenvalues and this resulted in a better condition number. The system we use is well-conditioned so it is less sensitive to perturbations of A or b. The growth of the solution was reduced as well.

The function that gives the value of the inner min for a given d was called innermin2. The optimal value of d is found by a maximization. This was done by minimizing -innermin2. To get an idea of the optimization landscape for this example, we plot z=-innermin2(d) for two dimensional d (or $n = 3$) in Figure 4.1.

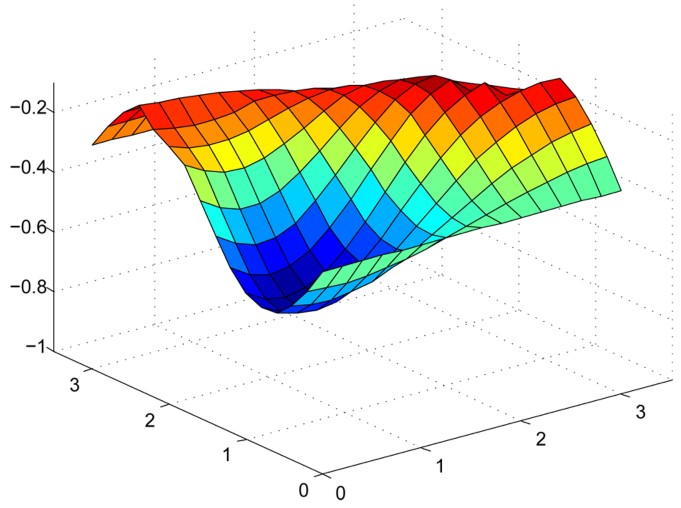

Figure 4.1: Plot of z=innermin2(d) for $n = 3$

We also plot v^* for $n = 1, 2, 3, 5$. Recall that $-v^*$ is proper if v^* is proper so for comparison purposes we actually plot the $-v^*$ for $n = 3, 5$ separately and then graph all of the test signals together. Table 4.1 gives the norm and coefficients of the optimal test signal found for each n.

Table 4.1: Example 7 Optimal Test Signal v Norm and Coefficients

n	norm	Coefficients				
1	5.4365	1	0	0	0	0
2	3.7456	0.6091	0.7931	0	0	0
3	3.6858	-0.5419	-0.8024	-0.2499	0	0
5	3.6247	-0.5172	-0.7795	-0.2795	-0.1631	-0.1418

Each n produces a suboptimal incipient test signal v^*. As we increase the number of parameters in v (the test signal form), the size of the test signal decreases. In this example, most of the improvement is present already by $n = 2$. This is consistent with some of the previous work on different problems where it was seen that on long time *intervals some type of sinusoid approaches optimal. Figure 4.6 is the result when all the test signals are plotted on the same graph.*

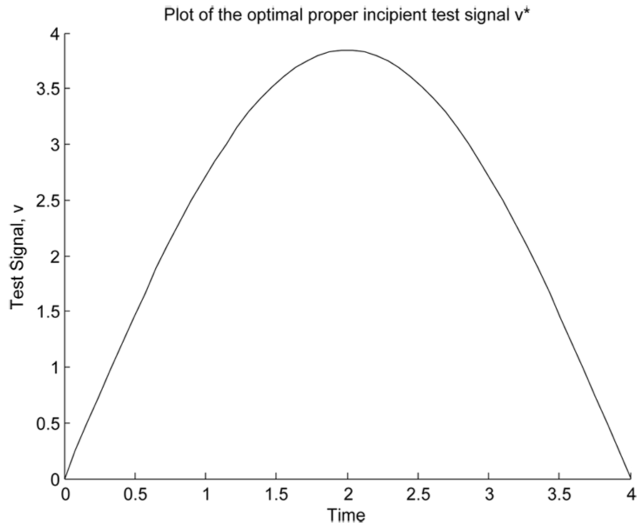

Figure 4.2: Optimal test signal for $n = 1$ for Example 7

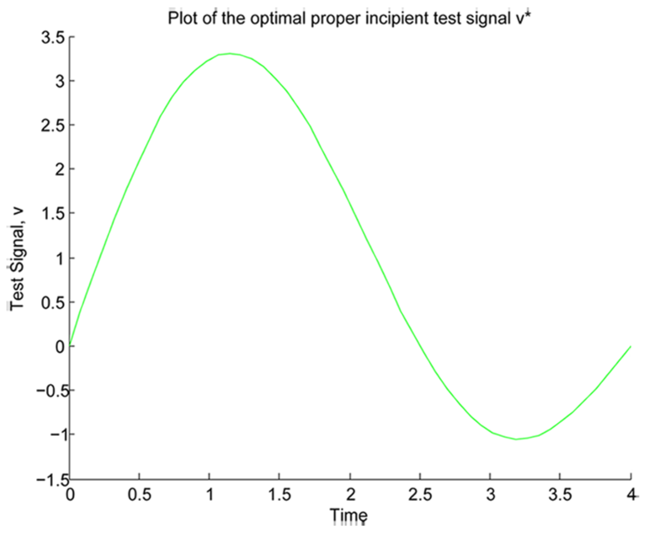

Figure 4.3: Optimal test signal for $n = 2$ for Example 7

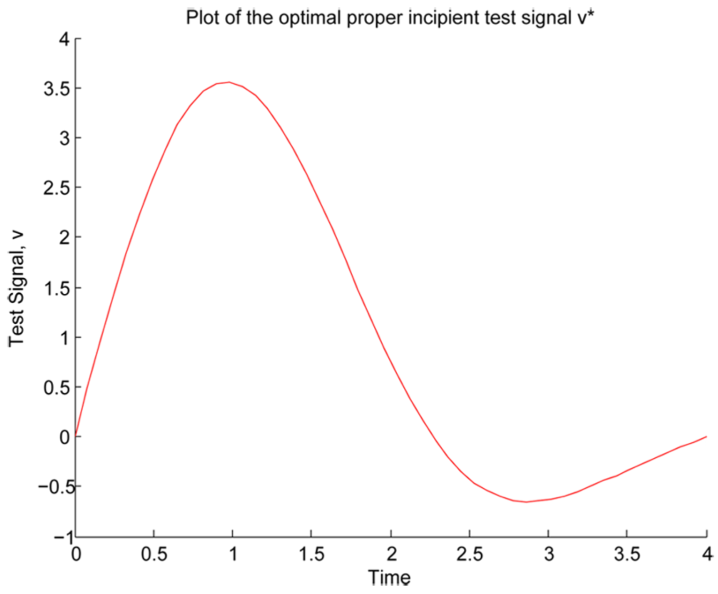

Figure 4.4: Optimal test signal for $n = 3$ for Example 7

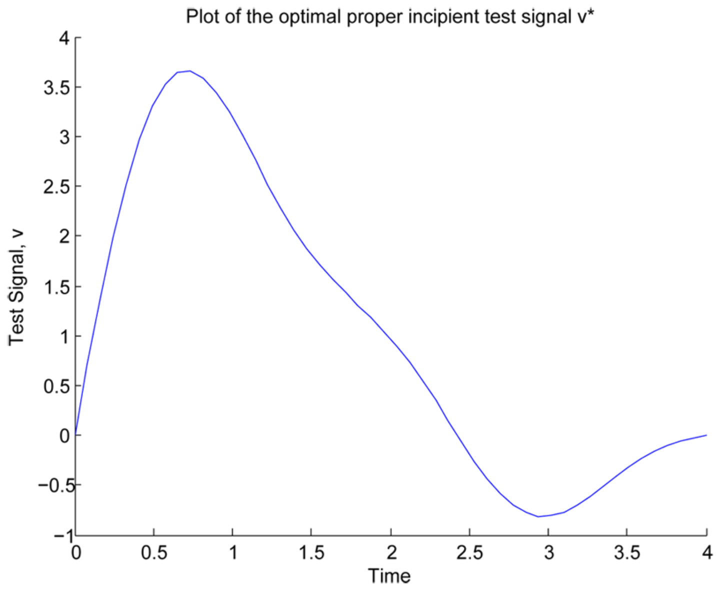

Figure 4.5: Optimal test signal for $n = 5$ for Example 7

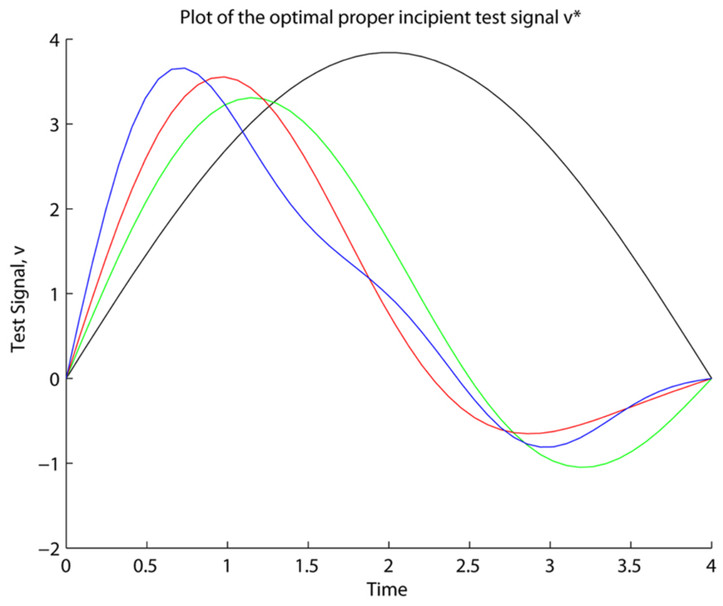

Figure 4.6: Optimal test signal for $n = 1, 2, 3, 5$ for Example 7

Example 8. *In the previous example both eigenvalues were real. Now we illustrate how different the problems can be when we consider a slightly different system matrix. Again using the procedure outlined in Section 4.5.1 (the same method implemented in Example 7), we examine the following system on the interval* $[0, 4]$

$$x' = \begin{pmatrix} -\theta_1 & -\theta_2 \\ 1 & -\theta_1 \end{pmatrix} x + \begin{pmatrix} 0 \\ 1 \end{pmatrix} v + \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \tag{4.108a}$$

$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} x + \nu_1. \tag{4.108b}$$

A change in θ_1 *may be thought of as a change in the internal damping of friction of the system while a change in* θ_2 *alters a frequency of the damped free response. We use the same finite dimensional family of test signals as in Example 7. The* $n = 1$ *case of equation (4.101) is also examined.*

The normal parameter values are $\theta_1 = 0.3$ *and* $\theta_2 = 4.0$ *so that the free response of the system is a lightly damped oscillation. The fault condition is given by* $\delta\theta_1 \in [-0.1, 0.1]$ *and* $\delta\theta_2 \in [1, 2]$*. Since the eigenvalues of this system are* $-\theta_1 \pm \sqrt{\theta_2}$ *we have that the problem is a lightly damped oscillation. We consider the perturbations as an uncertainty in the damping and an incipient fault in frequency.*

Table 4.2 gives the norm and coefficients of the optimal test signals found for each n*. Recall that* $-v^*$ *is proper if* v^* *is proper so for comparison purposes we actually plot the* $-v^*$ *for* $n = 3, 5$*. We plot* v^* *for* $n = 1, 2, 3, 5$*. Figure 4.11 gives the graph*

Table 4.2: Example 8 Optimal Test Signal v Norm and Coefficients

n	norm	Coefficients				
1	24.4291	1	0	0	0	0
2	8.5087	0.2061	0.9785	0	0	0
3	6.2633	0.5403	-0.3632	-0.7591	0	0
5	5.1781	0.0316	-0.3494	-0.8690	-0.3378	0.0881

of all of the test signals plotted together. It can be seen that the test signal for $n = 1$ *is larger than that for any* $n > 1$*. In fact just like in Example 7 as* n *increases the norm of* v *decreases and the inner minimum increases.*

This problem's solution is different from the solution of Example 7 in several ways. Here the extra terms in the approximation beyond $n = 2$ *continued to be useful. It is interesting to look at the optimal test signal. In this problem, the imaginary part of the eigenvalues for the nonfaulty system were* $\pm 2i$ *and for the faulty system were* $\pm \gamma i$ *with* $\sqrt{2} \leq \gamma \leq \sqrt{3}$*. Looking at the* $n = 5$ *case the dominant term in* v *of the form* $\sin(\psi t)$ *has* $\psi = 2.3562$*. Thus the optimizer chose a signal whose dominant frequency was both close to the frequency of the nonfaulty system yet some distance from the faulty frequencies.*

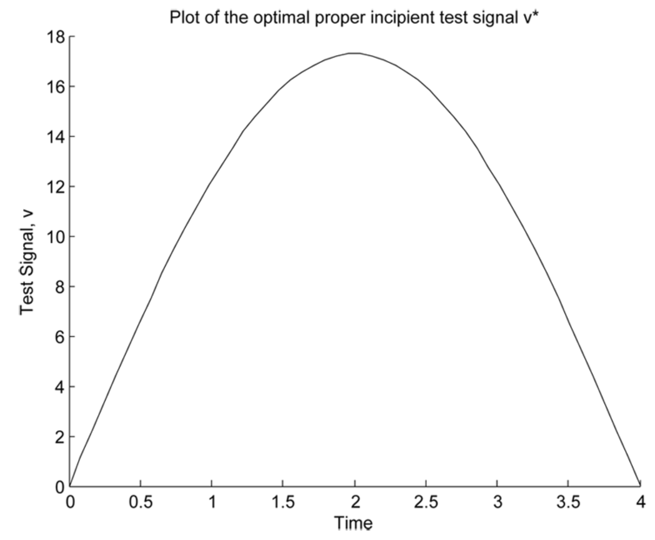

Figure 4.7: Optimal test signal for $n = 1$ for Example 8

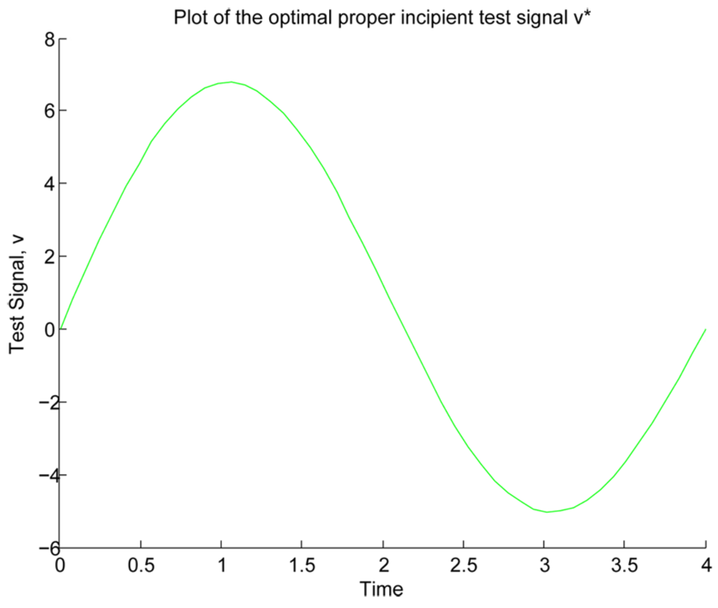

Figure 4.8: Optimal test signal for $n = 2$ for Example 8

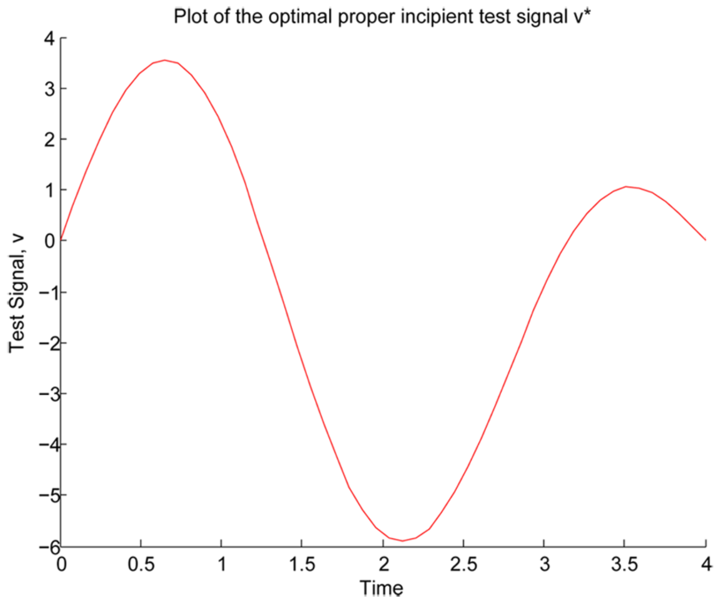

Figure 4.9: Optimal test signal for $n = 3$ for Example 8

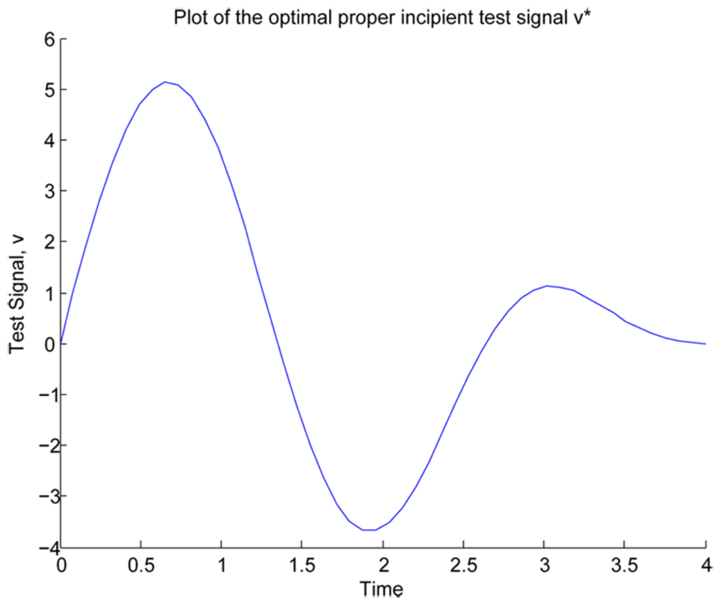

Figure 4.10: Optimal test signal for $n = 5$ for Example 8

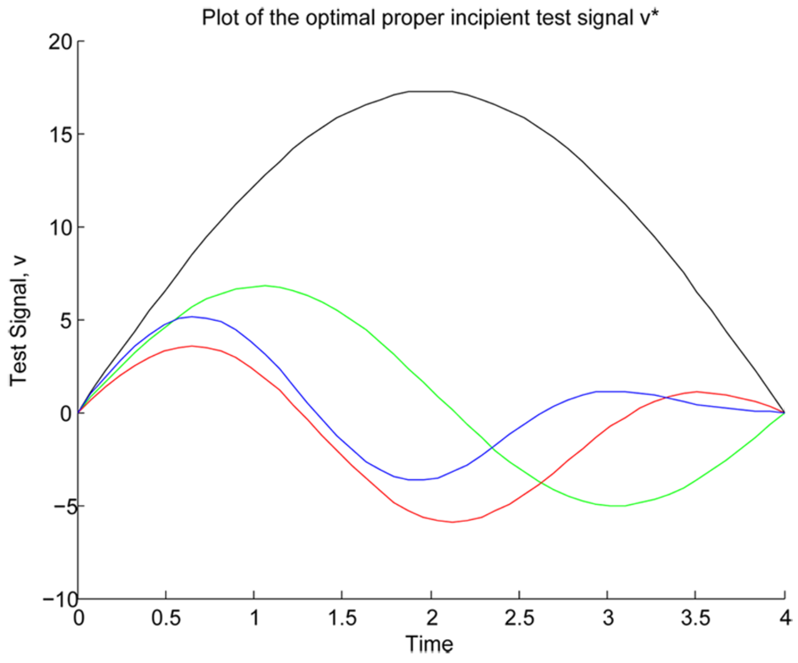

Figure 4.11: Optimal test signal for $n = 1, 2, 3, 5$ for Example 8

4.7 Using the Test Signal

In the standard two model test using an active test signal there is a direct procedure for using the test signal. Each model has a noise bound and as the output y is received, a filter is run which is computing the minimal noise needed to have y as an output of that particular model. Once one of the filters exceeds the bound, it is possible to decide if a fault is present or not. A similar idea can be used for this case, but we need to modify the method slightly because of the $\delta\theta$ parameter. Running a $\delta\theta$ optimization would be computationally expensive to have in the filter. Also, we use the two-norm on the total noise in the two models instead of the max norm as in [10]. These differences can be handled by noting that for linear problems with additive uncertainty that the test signals and noise bounds scale together. Note that for any pair of real numbers a, b we have

$$\max(\|a\|, \|b\|) \leq \sqrt{(a^2 + b^2)} \leq \sqrt{2}\max(\|a\|, \|b\|).$$

The actual application of the test signals would proceed as follows. Suppose that we have a set of incipient faults parameterized by Ω and given by (4.12). Assume that the noise bound for each model separately is

$$S_i = x_i(0)^T x_i(0) + \int_0^\omega \nu_i^T \nu_i + \mu_i^T \mu_i \, dt < \delta^2. \tag{4.109}$$

We compute the test signal v^* for a bound of 1. Let $\hat{v} = \delta\sqrt{2}v^*$. Using this test signal we have that if we get the same output from both the normal and any of the faulty models, then one of the models must have noise S_i larger than δ^2. We then run the standard filter on just the normal model. From Theorem 3.3.4 of [10] we have the following proposition.

Proposition 1: Given the systems (4.8), (4.9), and the noise bound (4.109). Let v^* be the test signal for the multi-model incipient problem found by the algorithm of this thesis with the bound $S < 1$. Let

$$\hat{v} = \delta\sqrt{2}v^*, Q = MM^T, \text{ and } R = NN^T.$$

Let

$$\gamma(s) = \textstyle\int_0^s \zeta^T R^{-1} \zeta dt$$

where

$$\begin{aligned} P' &= AP + PA^T - PC^T R^{-1} CP + Q, \quad P(0) = I && (4.110a) \\ z' &= Az - PC^T R^{-1} \zeta - B\hat{v} && (4.110b) \\ \zeta &= Cz - y. && (4.110c) \end{aligned}$$

If at any time τ during the test period we have $\gamma(\tau) > \delta^2$, then a fault has occurred. If $\gamma(\omega) \leq \delta^2$, then no fault has occurred.

P can be found either online or offline depending on the problem. If P is to be found online, then let

$$G_1 = C^T R^{-1} C, G_2 = R^{-1}, G_3 = C^T R^{-1}$$

and integrate the system

$$\begin{aligned} P' &= AP + PA^T - PG_1 P + Q, \quad P(0) = I && (4.111a) \\ z' &= (A - PG_2)z - PG_3 y - B\hat{v} && (4.111b) \\ \gamma' &= (Cz - y)^T R^{-1} (Cz - y), \gamma(0) = 0. && (4.111c) \end{aligned}$$

Note that (4.111) can be integrated as y arrives from the sensors so that $\zeta(t)$ is available in close to real time and the test can be stopped when the fault is detected.

4.8 Conclusions for the Continuous Time Case

The previous work on active failure detection has considered both additive and model uncertainty. While we have focused on additive uncertainty, we in fact are also including a type of model uncertainty. The model uncertainty considered here is quite different than that considered in [10] and the related papers. In [10] the model uncertainty is handled by writing a larger system using a formulation similar to that of [37]. While proven useful that type of model uncertainty has the effect in our setting of increasing the amount of uncertainty with increasing size of the test signal. In fact, very large test signals may not even be proper.

In this setting, we can think of part of Ω as the range of the incipient fault and part of it as a type of model uncertainty. Note that we only require that $0 \notin \Omega$, but for any particular variable θ_i we can allow $\delta\theta_i$ to vary over a set including 0. Thus we can use some of the $\delta\theta_i$ to express the incipient fault and the other $\delta\theta_j$ to express model uncertainty in either the faulty or nonfaulty model. Depending on how v and θ enter the equations it is no longer the case that there have to be limits on the size of the proper test signal. This can greatly increase the robustness of the numerical approaches used. Note that on Example 8 we allowed $\delta\theta_1$ to take on the value of zero.

While we do not discuss this in depth here, our results can also be useful in parameter identification. A test signal which is useful for testing an incipient fault in variable θ_i is also one for which the value of θ is sensitive to. This idea will be examined more carefully in chapter five.

This chapter presents the first algorithm for the computation of minimal proper test signals to detect multiple simultaneous incipient faults in continuous time linear systems. We have addressed the $\delta\theta_i$ as being parameter variations which are the incipient fault, but in fact part of them could be model uncertainty and a particular value of some of the drift parameters $\delta\theta_i$ could be zero. The only assumption that is required is that the vector perturbation $\delta\theta$ is bounded away from the origin. However, the computational algorithms and software may also need that Ω is convex in order to perform efficiently.

This work differs from the previous incipient work in several key ways. We allow for multiple faults to occur concurrently and we do not use linearizations, so there is not an assumption of small test signals. In addition, the model uncertainty is formulated in a very different way. For larger fault sets Ω, our signal is still guaranteed to be proper. In the previous work, the uncertainty appeared in the form of another type of noise and placed stronger restrictions on the test signal than we have with our approach here. Also, in previous studies larger test signals always expanded the amount of noise until at some point detection might be impossible. Finally, our approach to uncertainty is somewhat less conservative than the usual approximations which increase the allowed level of noise in order to incorporate the uncertainty.

Chapter 5

Future Work

We discuss some of the remaining open problems and highlight some practical applications related to the fault detection approach considered in this dissertation.

5.1 Examination of Model Uncertainty

A perfectly accurate mathematical model of a physical system can not be constructed. Usually, the parameters of the system vary with time and the characteristics of the noises are unknown so they cannot be modeled accurately. Therefore, there is always at least a slight difference between the actual process and its mathematical model even when there are no faults present. A good model should be simple enough to design, yet complex enough to give the engineer confidence that designs based on the model will work on the true plant [46].

The discrepancies that arise in design may cause difficulties in fault detection and isolation (FDI) applications. In particular, they may act as sources of false alarms or missed alarms. The modeling of uncertainties and noise is thus the most critical part in model-based FDI concepts and the solution to this problem is the key for its practical use. One way to address the problem of model uncertainty in a system is to create an FDI scheme that increases insensitivity to modeling uncertainty in order to provide increased fault sensitivity [43].

An important task of the model-based FDI scheme is to be able to identify incipient faults in a system. Unlike abrupt faults, incipient faults may have a small effect on residuals and they can be hidden by disturbances. The presence of incipient faults may not necessarily degrade the performance of the system, however, they may indicate that the component should be replaced before there is a chance of a more serious malfunction in the plant. As a result, the successful detection and diagnosis of incipient faults can be challenging in the design and evaluation of FDI algorithms [43].

Error and uncertainty are often used interchangeably to describe what we sometimes refer to as noise in an observed system. However, they are defined slightly differently. Error is defined as a recognizable deficiency in any phase or activity of modeling and simulation that is not due to lack of knowledge [1]. Uncertainty is defined as a potential deficiency in any phase or activity of the modeling process that is due to a lack of knowledge [1].

The definition for error implies that the deficiency is identifiable upon examination. Errors can also be classified as acknowledged or unacknowledged. Acknowledged errors (like round-off error and discretization error) have methods for identifying them and possibly removing them. Otherwise, they can remain in the code with their error estimated and listed. Unacknowledged errors (examples include computer programming errors or usage errors) have no set procedure for finding them and may continue within the code or simulation.

The key word in the definition of uncertainty is potential, which indicates that deficiencies may or may not exist. Lack of knowledge referenced in the definition has primarily to do with lack of knowledge about the physical processes that go into building the model. Sensitivity and uncertainty analyses can be used to better determine uncertainty.

There are two main types of uncertainty. This thesis deals with additive uncertainty and some types of model uncertainty. There is also the possibility of exploring model uncertainty further with this same type of problem.

Examining model uncertainty allows for the estimation of variability of the model parameters that result from random disturbances in the output. Understanding model variability helps with understanding how different the model parameters would be if the estimation was repeated using a different data set (with the same input sequence as the original data set) and the same model structure. Some references include [43, 46].

The parameter θ could be taken as model uncertainty. Based on [43], we consider the situation where the system matrices are functions of the parameter vector $\theta \in \Re^{n\times 1}$:

$$x_{k+1} = A(\theta)x_k + B(\theta)v_k + M\mu_k, \quad k = 0, \ldots, K-1 \tag{5.1}$$

If the parameter vector varies around the nominal condition $\theta = \theta_0$, then (5.1) can be rewritten as:

$$x_{k+1} = A(\theta)x_k + B(\theta)v_k + M\mu_k + \sum_{i=1}^{g}(\frac{\partial A}{\partial \theta_i}\delta\theta_i x + \frac{\partial B}{\partial \theta_i}\delta\theta_i v), k = 0, \ldots, K-1 \tag{5.2}$$

In this case, the distribution matrix and unknown input vector can be represented by:

$$E = \left(\tfrac{\partial A}{\partial \theta_1}| \quad \tfrac{\partial B}{\partial \theta_1}| \quad \ldots| \quad \tfrac{\partial A}{\partial \theta_g}| \quad \tfrac{\partial B}{\partial \theta_g}\right), d(t) = \left(\delta\theta_1 x^T| \quad \delta\theta_1 v^T| \quad \ldots| \quad \delta\theta_g x^T| \quad \delta\theta_g v^T\right)^T.$$

5.2 Fault Identification

A fault is defined as an unacceptable deviation of the system behavior. It is a malfunction that can disturb the normal operation of a system, causing an unacceptable deterioration of the performance of the system. Therefore, it is important to diagnose faults as early as possible [45].

Fault diagnosis consists of determining the fault type with as many details as possible such as the fault size, location, and time of detection. Fault diagnosis usually includes the following tasks:

- Fault detection - the existence of a fault, which leads to undesirable behavior in a system, determined;
- Fault isolation - the location of the fault (which component has become faulty) is determined;
- Fault identification - the magnitude, type, and cause of the fault are estimated.

In this thesis we focus on detecting faults only, but there is also the problem of fault identification which is considered the most important of all of the fault diagnosis tasks [43]. The detection and isolation of a fault are more easily achieved during the diagnosis process so there have been many studies on these topics. However, the fault identification problem itself has not gained as much research attention [43].

5.3 Parameter Identification

There is another problem called parameter identification. Parameter identification is an important part of any successful design for controlling systems with unknown parameters [35]. For continuous linear time invariant systems, the Kalman filter-based identifier is a popular design method for online parameter identification. The Kalman filter-based identification scheme can be applied directly to the system when the state variable and its derivative are both available for identification purposes. However, if the only information we have available to us is noise corrupted output measurements, the identification scheme must utilize the Kreisselmeier observer in the single-input-single-output (SISO) case or a general prefiltering-based design for the multi-input-multi-output (MIMO) case. After this process, the problem is converted to one that can be solved using the Kalman filter-based method [35].

One can think of our design as determining a test signal v for which the output is more sensitive to parameter change. This suggests our results might be useful also for designing test signals for parameter identification.

5.4 More Efficient Algorithm

When writing code to solve a problem it is important to consider the efficiency of the algorithm used. One major factor to consider when analyzing the efficiency of an algorithm is the time it takes to get a solution. It takes about half an hour to run the code to solve a two parameter problem for $n > 2$ for Examples 7 and 8 in Chapter 4. Accordingly, developing code that solves the continuous time case faster is a topic for future study.

5.5 Applications

Incipient fault detection consists of early detection of small variations in system behavior. Therefore this technique can be very useful in maintaining the integrity and efficiency of practical systems. In addition, there is usually a cost savings when potential machine failures are detected before they occur. For example, in manufacturing, machine tools are used extensively to create objects to fit specific needs. Recent studies indicate that as many as ninety percent of the failures of machine tools occur because of the malfunction of internal components such as the main motor. Therefore, it is important that these motor malfunctions are detected and corrected before the quality of the system is degraded and the overall system is jeopardized [13]. Some of the applications include detecting incipient faults in small DC motors found in personal computers, big motors used in power plants, and valve actuators found in HVAC systems.

Future research would include the investigation of specific applications and efficient algorithms tailored to these applications.

Chapter 6

Concluding Remarks and Contributions

6.1 Concluding Remarks

We assume that the detection signal v is available to us over a given time interval and we use it to reveal the presence of incipient faults. There are multiple questions that must be addressed in finding solutions to this optimal control problem: under what conditions does an input v exist such that small parameter variations (incipient faults) of a given size can be detected, what is the smallest energy v resulting in detection, and how can the detection procedure be implemented on-line. It should be noted that detection is considered possible if the set of outputs y corresponding to the nominal system and the set of outputs corresponding to the faulty system do not intersect.

We use an active approach to detect incipient faults in linear systems where there is more than one drifting parameter. A threshold is specified and we seek detection if some combination of uncertainties is exceeded. The shape of a good test signal independent of $\delta\theta$ is determined. Then, we scale the detection signal depending on the specified threshold. The incipient test signal solution can be compared to the solutions found using the two model approach.

In the two model approach, we find a test signal for two fixed models that guarantees detection for a special case. The incipient test signal is found using two models that are the result of problem reformulation and a limiting process. Therefore, incipient test signals can be very different from the two model test signals. With scaling, the incipient detection signal is designed to work for a parameterized family of models.

If there is only one $\delta\theta$, then we get two models just like the model problem. However, the value for $\delta\theta$ may vary and we consider a few different values. As a result, we do not want to compute a separate v for every value of $\delta\theta$. Therefore, we construct a model where θ is the parameter and another model where $\theta + \delta\theta$ is the parameter ($\delta\theta \neq 0$). We compute a signal that can work for a range of values of $\delta\theta$.

In the process, we do some approximations and a change of variables. This gives equations that look like two models with no $\delta\theta$ in the expression. There is a new optimal test signal, $v^*_{inc} = \frac{1}{\delta\kappa}\frac{1}{\zeta}w^*$, where we take $w^* = \frac{1}{\sigma}\gamma$ for the discrete time case. We solve for $\tilde{v}$ given $\delta\kappa$ using $\frac{\tilde{v}}{\delta\kappa}$. $\tilde{v}$ is actually a suboptimal test signal due to approximation. In certain places, we drop higher order terms. Since this is an approximation, this works if $\delta\theta$ is small enough. If the model is linear, then we will not get the higher order terms so the approximation will be the actual result. However, with the nonlinear case, we must be more careful because we will lose some higher order terms when calculating the approximation. As a result, for the linear case, we can use a larger interval for $\delta\theta$ values, but in the nonlinear case the model works for smaller intervals of $\delta\theta$. The continuous case works in a similar manner.

We found that for the discrete time case

- v^*_{inc} covers a range of problems, so we expect it to be larger than just one test signal for a special case
- For smaller $\delta\kappa$, v^*_{inc} is closer to the solution of the two model test signals
- The problem solved is a min-max problem

Discrete systems can arise in different ways. They can be discrete or they can be approximations of continuous time systems. Example 6, which is set up as a discrete problem, is an approximation of a continuous time system. In fact, it is what happens when you apply Euler's numerical method to the equation $x'' + dx' + sx = v$.

In general, we sometimes approximate a continuous time system by a discrete system. However, if there are lots of time steps it is sometimes better to look at a discrete time system as continuous. However, this is only possible sometimes. In addition, depending on the algorithm implemented to solve for the incipient test signal it may be better to choose either a discrete time or continuous time model.

6.2 Contributions

6.2.1 Papers

- *Active Incipient Fault Detection With Two Simultaneous Faults*

  - Safe Process 2009 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Barcelona, Spain, presenter Alireza Esna Ashari

  This paper addresses the problem of detecting small parameter variations in linear uncertain systems due to incipient faults by injecting an input signal to enhance detection of the faults. Previous work assumed that there was only one fault occurring at a time. We allow for two concurrent faults, which is a natural assumption in the incipient case. A useful method for the construction of an optimal input signal for achieving guaranteed detection with specified precision is presented in the discrete model case. This method is an extension of the multi-model approach used for the construction of auxiliary signals for failure detection, however, new technical issues included. The paper was presented.

- *Active Incipient Fault Detection With More Than Two Simultaneous Faults*

  - 2009 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, Texas

  This paper extends the idea highlighted in *Active Incipient Fault Detection With Two Simultaneous Faults* by focusing on the case where we look at linear uncertain systems with three incipient faults occurring simultaneously. A computational method for the construction of an input signal for achieving guaranteed detection with specified precision is presented for discrete time systems. This method is an extension of the multi-model approach used for the construction of auxiliary signals for failure detection, however, new technical issues included. A case study is examined.

- *Active Incipient Fault Detection In Continuous Time Systems With Multiple Simultaneous Faults*

  - Provisionally Accepted to Numerical Algebra, Control, and Optimization Journal in 2010

  The problem of detecting small parameter variations in linear uncertain systems due to incipient faults, with the possibility of injecting an input signal to enhance detection is considered. This paper extends the work of *Active Incipient Fault Detection With More Than Two Simultaneous Faults* to continuous time systems. For the continuous time incipient fault problem the information from the discrete time case that is relevant is utilized. However, a different method is implemented to calculate the optimal incipient test signal for the continuous time case. An LQR problem is solved in order to find a solution for the test signal. A case study is examined.

6.2.2 Presentations

- *Active Incipient Fault Detection With More Than Two Simultaneous Faults*, IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX, October 2009.

- *Active Incipient Fault Detection With Multiple Concurrent Faults*, SIAM-SEAS Conference, Raleigh, NC, March 2010.
- *Active Incipient Fault Detection With Multiple Simultaneous Faults (Continuous Time Case)*, SIAM National Meeting, Pittsburgh, PA, July 2010.

Bibliography

[1] American Institute of Aeronautics and Astronautics (1998) *Guide for the verification and validation of computational fluid dynamics simulations.* AIAA G-077-1998.

[2] Antsaklis, P.J. and Michel, A.N. (1997) *Linear Systems.* McGraw-Hill, New York.

[3] Ashari, A.E., Nikoukhah, R. and Campbell, S. (2010) *Active robust fault detection in closed-loop systems: quadratic optimization approach.* IFAC Symposium, Spain.

[4] Ashari, A.E., Nikoukhah, R. and Campbell, S. (2010) *Effects of feedback on active fault detection.* Submitted.

[5] Ashari, A.E., Nikoukhah, R. and Campbell, S. (2010) *Auxiliary signal design for robust active fault detection of linear discrete-time systems.* Submitted.

[6] Ashari, A.E., Nikoukhah, R. and Campbell, S. (2010) *A numerical method to solve a quadratic constrained maximization.* Submitted.

[7] Ashari, A.E., Nikoukhah, R. and Campbell, S. (2009) *Feedback in active fault detection.* Proc. IEEE Conference on Decision and Control, Shanghai, China.

[8] Ashari, A.E., Nikoukhah, R. and Campbell, S. (2009) *Robust multi-model fault detection and control using an active closed-loop approach.* Proc. SYSID, St. Malol, France.

[9] Campbell, S.L. and Meyer, C. (2008) *Generalized Inverses of Linear Transformations.* SIAM Publishers, Philadelphia, PA.

[10] Campbell, S.L. and Nikoukhah, R. (2004) *Auxiliary Signal Design for Failure Detection.* Princeton University Press, Princeton.

[11] Choe, D., Campbell, S.L., and Nikoukhah, R. (2005) *A comparison of optimal and suboptimal auxiliary signal design approaches for robust failure detection.* IEEE Conference on Control Applications, Toronto.

[12] Choe, D., Campbell, S.L., and Nikoukhah, R. (2009) *Optimal piecewise-constant signal design for active fault detection.* International Journal of Control, 82, 130–146.

[13] Chow, M.-Y. (1997) *Methodologies of Using Neural Network and Fuzzy Logic Technologies for Motor Incipient Fault Detection.* World Scientific Publishing Company.

[14] Demetriou, M.A. and Polycarpou, M.M. (1998) *Incipient fault diagnosis of dynamical systems using on-line approximators.* IEEE Trans. Automatic Control, 43, 1612–1617.

[15] Drake, K.J., Campbell, S.L., Andjelkovic, I., and Sweetingham, K. (2006) *Active incipient failure detection: a nonlinear case study.* Proc 4th International Conference on Computing, Communications and Control Technologies, Orlando, FL.

[16] Fair, M. and Campbell, S.L. (2009) *Active incipient fault detection with two simultaneous faults.* SafeProcess 2009 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Barcelona, Spain.

[17] Fair, M. and Campbell, S.L. (2009) *Active incipient fault detection with more than two simultaneous faults.* IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, TX.

[18] Fair, M. and Campbell, S.L. (2010) *Active incipient fault detection in continuous time systems with multiple simultaneous faults.* Submitted to Numerical Algebra, Control, and Optimization Journal.

[19] Goode, P.V. and Chow, M.-Y. (1995) *Using a neural/fuzzy system extract knowledge of incipient fault in induction motors part I - methodology.* IEEE Trans. Industrial Electronics, 42, 131–138.

[20] Goode, P.V. and Chow, M.-Y. (1995) *Using a neural/fuzzy system extract knowledge of incipient fault in induction motors part II - application.* IEEE Trans. Industrial Electronics, 42, 139–146.

[21] Higham, D.J. and Higham, N.J. (2005) *Matlab Guide Second Edition.* SIAM Publishers, Philadelphia, PA.

[22] Horton, K. (2001) *Fault detection and model identification in linear dynamical systems.* PhD thesis, North Carolina State University.

[23] Iserman, R. (2006) *Fault Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance.* Springer, Berlin.

[24] Jiang, B., Staroswiecki, M., and Cocquempot, V. (2003) *Active fault tolerant control for a class of nonlinear systems.* Proc. Safeprocess, Washington, D.C., 127–132.

[25] Kerestecioglu, F. and Zarrop, M.B. (1994) *Input design for detection of abrupt changes in dynamical systems.* Int. J. Control 59: 1063–1084.

[26] Lewis, F.L. and Syrmos,V.L. (1995) *Optimal Control.* Wiley-Interscience, New York.

[27] Meyer, C. (2000) *Matrix Analysis and Applied Linear Algebra.* SIAM, Philadelphia, PA.

[28] Niemann, H.H. (2006) *A setup for active fault diagnosis.* IEEE Transactions on Automatic Control, 51:1572–1578.

[29] Niemann, H.H. (2006) *Active fault diagnosis in closed-loop uncertain systems.* In Preprints of 6th IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes, SAFEPROCESS'2006, Beijing.

[30] Nikoukhah, R. and Campbell, S.L. (2006) *Auxiliary signal design for active failure detection in uncertain linear systems with a priori information.* Automatica, 42: 219–228.

[31] Nikoukhah, R. and Campbell, S.L. (2008) *On the detection of small parameter variations in linear uncertain systems.* European J. Control, 14, 158–171.

[32] Nikoukhah, R. and Campbell, S.L. (2006) *Robust detection of incipient faults: an active approach.* Proc. IEEE Med. Conf. Control and Automation.

[33] Nikoukhah, R., Campbell, S.L., and Drake K. (2008) *An active approach for detection of incipient faults.* International Journal of Systems Sciences, 41, 2: 241–257.

[34] Nikoukhah, R., Campbell, S.L., Horton, K., and Delebecque, F. (2002) *Auxiliary signal design for robust multi-model identification.* IEEE Transactions on Automatic Control, 38:8, 1313–1325.

[35] Pan, Z. and Basar, T. (1996) *Parameter identification for uncertain linear systems with partial state measurements under an H^∞ criterion.* IEEE Transactions on Automatic Control, 41, 9: 1295–1311.

[36] Patton, R.J., Frank, P.M., and Clark, R.N. (2000) *Issues of Fault Diagnosis for Dynamic Systems.* Springer, Berlin.

[37] Petersen, I.R. and McFarlane D.C. (1999) *A methodology for process fault detection.* IEEE Conference on Decision and Control, Phoenix, AZ, 4984–4989.

[38] Savkin, A.V. and Petersen, I.R. (1997) *A new approach to model validation and fault diagnosis.* J. Optimization Theory and Application, 94, 241–250.

[39] Shampine, L.F., Gladwell, I., and Thompson, S. (2003) *Solving odes with Matlab.* Cambridge University Press.

[40] Shampine, L.F., Kierzenka, J., and Reichelt, M.W. (2000) *Solving boundary value problems for ordinary differential equations in Matlab with bvp4c.* Manuscript, available at http://www.mathworks.com/bvptutorial, 2000.

[41] Simandl, M., Puncochar, I., and Kralovec, J. (2005) *Rolling horizon for active fault detection.* Proc. of IEEE Conf. on Decision and Control and European Control Conf. Seville, Spain, 3789–3794.

[42] Simandl, M., and Puncochar, I. (2007) *Unified solution of optimal active fault detection and optimal control.* Proc. 2007 American Control Conf. New York, USA, 3222–3227.

[43] Simani, S., Fantuzzi, C., and Patton, R.J.(2003) *Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques.* Springer, Great Britain.

[44] Staroswiecki M. and Jiang, B. (2001) *Fault identification for a class of linear systems based on adaptive observer.* Proc. of 40th IEEE Conf. on Decision and Control.

[45] Zhou, Y. and Dexter, A. (2009) *Estimating the size of incipient faults in HVAC equipment.* HVAC and R Research, 15:1, 151–163.

[46] Zhou, K., Doyle, J.C., and Glover, K. (1996) *Robust And Optimal Control.* Prentice-Hall, New Jersey.

Appendices

Definitions

Actuator is a mechanical device for moving or controlling a mechanism or system; a mechanical device that puts something into automatic action. It takes energy, usually transported by air, electric current, or liquid and converts that into some kind of motion.

Condition number The condition number of a matrix A is the quantity $\|A\|\|A^{-1}\|$. It is the measure of sensitivity of the solution $Ax = b$ to the perturbations of A or b. If the condition number is one, A is said to be perfectly conditioned. A problem with a low condition number (relative to one) is said to be well-conditioned. If the condition number of A is large, A is said to be ill-conditioned.

Degenerate is a limiting case of some type of entity that is equivalent to some simpler type, often obtained by setting some coefficient or parameter to zero. For instance, when using an optimizer if the bounds have been set such that the optimal point is outside of the boundary and the initial guess is close to a certain value, then we may get a degenerate point.

Deterministic system is a system in which the output can be predicted with 100 percent certainty.

Disturbance is an unknown and uncontrolled input acting on a system.

Fault is an unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable, usual, or standard condition.

Failure is a permanent interruption of a system's ability to perform a required function under specified operating conditions.

Failure detection is the early recognition of problem-prone behavior in an observed system in order to prevent the system from shutting down or causing a problematic incident. Failure detection increases system safety, reliability, and availability.

Fault diagnosis consists of determining the type, size, and location of the most possible fault, as well as the time of detection. Fault diagnosis procedures use the analytic and heuristics/human knowledge (that is, a set of rules intended to increase the probability of solving some problem) symptoms.

Fault identification is the determination of the size and time-invariant behavior of a fault.

Fault tolerance means faults are compensated in such a way that they do not lead to system failures. Unavoidable faults should be tolerated by additional design efforts, like additive uncertainty in our models.

Feedback control is a type of system control we get when a part of the output signal is operated upon and fed back to the input in order to obtain a desired effect. It is the measurement of differences between planned outputs and actual outputs achieved, and the modification of subsequent action and/or plans to achieve future required results.

Fourier Series decomposes any periodic function or periodic signal into the sum of a (possibly infinite) set of simple oscillating functions, namely sines and cosines (or complex exponentials). The Fourier series has many applications in electrical engineering, optics, signal processing, etc.

Hybrid systems are systems modeled by discrete dynamics combined with continuous dynamics. Also, although most systems are usually described by either a continuous-time model or a discrete-time model, when designing a stabilizing feedback, it may be necessary (or more efficient) to consider a hybrid feedback law. In this context, the system in closed loop becomes hybrid.

Ill-conditioned An ill-conditioned matrix is one where the solution to $Ax = b$ is overly sensitive to perturbations in A or b.

Least Squares Solution According to Wolfram Online, if we want to find solutions to the matrix equation $Ax = B$, where A is an mxn matrix and $m > n$, there may be no solution. In these cases, it is possible to find a best fit solution that minimizes $\|Ax - B\|_2$ (two-norm). This generates a least squares solution. This is often used because the function $\frac{1}{2}\|Ax - B\|_2^2$ is differentiable in x and the two-norm is preserved under orthogonal transformations.

If the rank of A is n (full column rank),it can be shown that there is a unique solution to the least squares problem and it solves the linear system. If A is an $m \times n$ matrix with $m > n$ (i.e., there are more equations than unknowns), then the general way to find a least squares solution to this overdetermined system is to use SVD to form a matrix known as the pseudo-inverse of the matrix A. We want to solve $Ax = B$, so we find $p = A^{\dagger}$ (pseudo-inverse). Then, $x = pB$. This technique works even if the input matrix A is rank deficient.

Max norm also called the infinity norm is defined by $\|x\|_{\infty} = \max_i |x_i|$.

Nondeterministic system is a system in which the output cannot be predicted because there are multiple possible outcomes for each input.

Optimization In mathematics, optimization or mathematical programming, refers to choosing the best element from some set of available options. In the simplest case, this means solving problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables from within an allowed set. This scalar real valued objective (cost) function is actually a small subset of this field which consists of a large area of applied mathematics. In general, it means finding 'the best values' of some objective function given a defined domain, including a variety of different types of objective functions and different types of domains.

Orthonormal set $B = u_1, u_2, \ldots, u_n$ is called an orthonormal set whenever each $\|u_i\| = 1$ for each i, and $u_i \perp u_j$ for all $i \neq j$. In other words, the inner product of u_i, u_j is equal to one when $i = j$ and zero when $i \neq j$. Every orthonormal set is linearly independent. Every orthonormal set of n vectors from an n-dimensional space V is an orthonormal basis for V. See pages 298-300 of [27] for more about orthonormal bases and **Fourier Expansions**.

Parameter is a factor that determines a range of variations; it creates a boundary in a problem. It can also be defined as a constant or variable term in a function that determines the specific form of the function, but not its general nature, as b in $f(x) = bx$, where b determines only the slope of the line described by $f(x)$.

Pseudo inverse is a matrix inverse that may be defined for a complex matrix, even if it is not square. It is possible to construct many different pseudo inverses for the same matrix; the pseudo inverse is not unique. The most commonly used pseudo inverse, which will be used in this work, is the Moore-Penrose pseudo inverse.

Rank Let $A_{m\times n}$ be a matrix. Then, rank(A^*)=rank(A). Also, rank(A) is the maximum number of linearly independent rows.

Sensor is a device that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument. In other words, a sensor is a device which receives and responds to a signal. A sensor's sensitivity indicates how much the sensor's output changes when the measured quantity changes. Applications include cars, machines, manufacturing, and robotics.

Singular Value Decomposition (SVD) For each $A_{m\times n}$ matrix of rank r, there are orthogonal matrices $U_{m\times m}$ and $V_{n\times n}$ ($U^TU = I, V^TV = I$) and a diagonal matrix $D_{r\times r} = Diag(\sigma_1, \sigma_2, ..., \sigma_r)$ such that

$$A = U \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} V^T$$

with $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$. The σ_i's are called the nonzero singular values of A. When $r < p$, where $p = min(m, n)$, A is said to have $p - r$ additional zero singular values. This factorization is called a SVD of A, the columns of U and V are called left-hand and right-hand singular vectors for A, respectively. The σ_i's are the eigenvalues of A^*A and the singular vectors are specialized sets of eigenvectors for A^*A.

SVD Example-Wikipedia

Let $H_{4\times5}$ be a matrix such that there is a SVD $H = U\Sigma V^T$, where $U_{4\times4}, V_{5\times5}$ are both orthogonal matrices and $\Sigma_{4\times5}$ with $D_{r\times r}$, r=rank of H.

V contains an orthonormal set of input basis vectors for H and U contains an orthonormal set of output basis vectors for H. Also, Σ contains the singular values of H since σ_i^2's are actually eigenvalues of H^TH. Suppose $U = (U_1, U_2)$, then U_1 is an orthonormal basis for $R(H)$ (first r columns) and U_2 is an orthonormal basis for $N(H^T)$ (last $m - r$ columns). Let $V = (V_1, V_2)$, where V_1 is an orthonormal basis for $R(H^T)$ (first r columns) and V_2 is an orthonormal basis for $N(H)$ (last $n - r$ columns).

Σ^{-1} is the transpose of Σ with every nonzero entry replaced by its reciprocal. $H^T H$ is numerically unstable, especially for singular values close to zero.

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \end{pmatrix}$$

Then, an SVD of H represented by $H = U\Sigma V^T$ is given by

$$H_{4\times 5} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{5} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \sqrt{.2} & 0 & 0 & 0 & \sqrt{.8} \\ 0 & 0 & 0 & 1 & 0 \\ -\sqrt{.8} & 0 & 0 & 0 & \sqrt{.2} \end{pmatrix}$$

Hence, we can determine $H^\dagger = V\Sigma^{-1}U^T$

$$H^\dagger_{5\times 4} = \begin{pmatrix} 0 & 0 & \sqrt{.2} & 0 & -\sqrt{8} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \sqrt{.8} & 0 & \sqrt{.2} \end{pmatrix} \begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{5}} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

which is the pseudo inverse of H.

SVD Command in Matlab $[U, S, V] = svd(X)$ produces a diagonal matrix S, with the same dimension as X, containing nonnegative diagonal elements in decreasing order (these are the singular values of X), and unitary matrices U and V so that $X = USV^*$. The columns of V are the right-singular vectors of X for the corresponding singular values (found on the diagonal of S). Therefore, the rows of V^* are right-singular vectors that correspond to the singular values of X.

Static case is the case when the processes have no dynamics and are simply functions between vector spaces.

Unitary and Orthogonal Matrices These types of matrices have some nice features, one of which is the fact that they are easy to invert. The columns of $U_{m\times m} = (u_1, u_2, ..., u_m)$ are an orthonormal set meaning $[U * U]_{ij} = u_i^* u_j$ (inner product) and this equals 1 when $i = j$ and it equals 0 when $i \neq j$. Also, $U^*U = I$ which implies that $U^{-1} = U^*$. Since $U^*U = I$ iff $UU^* = I$, the columns of U are orthonormal iff the rows of U are orthonormal. As a result, the definition of these matrices can be stated either in terms of orthonormal columns or orthonormal rows. Recall: Every orthonormal set is linearly independent. Further, multiplication by an orthogonal matrix does not change the length of the vector. Say U is an orthogonal matrix, then $\|Ux\|^2 = \|x\|^2$ for any vector $x \in \Re^n$.

Well-conditioned means numerically stable; having a small condition number.

Matlab Commands Used

Matlab Integrator

Ordinary differential equations (odes) arise in many mathematical models and describe phenomena that continuously change. Therefore a system of odes has many solutions, but often there is a particular solution of interest. This solution is determined by specifying the values of all of the system's components at a single point, $x = a$. This is called an initial value problem (IVP).

However, in many applications a solution is determined in a more complex way. A boundary value problem (BVP), like the one solved in this thesis, specifies values or equations for solution components at more than one x. Unlike IVPs, a BVP may not have a solution, or it may have a finite number of solutions, or it may have infinitely many solutions. As a result, programs written to solve BVPs require users to provide an initial guess for the solution desired. Sometimes parameters also need to be determined.

bvp4c

bvp4c solves boundary value problems for ordinary differential equations. The general syntax for this particular function is as follows: sol=bvp4c(odef, bcf, solinit), solinit = bvpinit(x, yinit, params).

odef is a function handle that evaluates the ode $f(x, y)$, which can have one of two forms: dydx=odef(x, y), dydx=odef(x, y,params), where x is a scalar, y is a column vector, and params represents a vector of unknowns. The output dydx is a column vector.

bcf is a function handle that computes the residual in the boundary conditions. For two-point boundary value conditions of the form bc($y(a), y(b)$), bcf can have the form: res=bcf(x, y), res=bcf(x, y,params). x, y, and params are defined as above.

$y(a)$ corresponds to initial conditions, $y(b)$ corresponds to final conditions and res is a column vector. There is also the possibility of having a multi-point boundary value problem.

solinit is a structure containing the initial guess for a solution. The user must create the function bvpinit.solinit which includes the following fields: x, y, and params. x represents the ordered nodes of the initial mesh. The boundary conditions are imposed at the initial and final times. This y is the initial guess for the solution such that solinit.y(:,i) is a guess for the solution at the node solinit.x(i). params is optional and it provides an initial guess for unknown parameters. The structure can have any name, but the fields must be named x, y, and parameters.

sol=bvp4c(odef,bcf,solinit) integrates a system of ordinary differential equations of the form $y' = f(x, y)$ on the interval $[a, b]$ subject to two-point boundary value conditions $bc(y(a), y(b)) = 0$.

Matlab Optimizers

In general, these are the steps to take in order to use the optimizers in Matlab. Create a separate M-file that only has the objective function in it. Use the handle sign (@) in the optimization function call to let Matlab know this is the function to optimize. Also, include the initial starting point and any other parameters for the function. This is the method we implement using fmincon.

Generally, positive exitflag values are good. Usually the larger the exitflag value, the more confident the optimizer is that it has found a good result.

fmincon

This optimizer tries to find the minimum of a constrained nonlinear multivariable function starting at an initial estimate. This is usually referred to as constrained nonlinear optimization or nonlinear programming.

[x,fval,exitflag] = fmincon(fun,$x0, A, b, Aeq, beq$,lb,ub,nonlcon,options) minimizes with the optimization options specified in the structure options. x minimizes the objective function fun, fval returns the value of the objective function fun at the solution x, and exitflag gives a numerical value that indicates the integrity of the optimization. Optimset can be used to set the structure options. If there are no nonlinear inequality or equality constraints, set nonlcon = []. Here $x0$ is the initial estimate, A, b, Aeq, beq are used for any linear constraints in the problem, where $Ax \leq b$ or Aeq x = beq if it exists. Also, lb represents the lower bound of x and ub stands for the upper bound of x if it exists. If any of these constraints are not present in the optimization problem, then instead of a value or function, you put [] to indicate it does not exist.

Optimization Options

Some options apply to all algorithms and others are relevant only to certain algorithms. Optimset can be used to set or change the value of these fields in structure options.

fmincon uses one of following three algorithms: active-set, interior-point, or trust-region-reflective. You choose the algorithm at the command line with optimset. For example: options=optimset('Algorithm','active-set').

The default trust-region-reflective (formerly called large-scale) requires: a gradient to be supplied in the objective function, 'GradObj' to be set to 'on', and either bound constraints or linear equality constraints, but not both. If these conditions are not all satisfied, the 'active-set' algorithm (formerly called medium-scale) is the default. The 'active-set' algorithm is not a large-scale algorithm.

Exitflag

If the value of the exitflag is greater than 0 this means that fmincon converged to a solution. If the exitflag equals 0, this means that the maximum number of function evaluations was reached. If the exitflag is less than 0, then fmincon did not converge to a solution.

There are eight possible exitflag messages in fmincon. 1-first order optimality measure was less than options.TolFun and maximum constraint violation was less than options.TolCon, 2- the change in x was less than options.TolX, 3-the change in the objective function value was less than options.TolFun, 4-the magnitude of the search direction was less than 2*options.TolX and constraint violation was less than options.TolCon, 5-the magnitude of directional derivative in search direction was less than 2*options.TolFun and maximum constraint violation was less than options.TolCon, 0-the number of iterations exceeded options.MaxIter or number of function evaluations exceeded options.FunEvals, -1-the output function terminated the algorithm, or -2-no feasible point was found.

Limitations

fmincon is a gradient-based method designed to work on problems where the objective and constraint functions are both continuous and have continuous first derivatives.

When the problem is infeasible, fmincon attempts to minimize the maximum constraint value. The trust-region-reflective algorithm does not allow equal upper and lower bounds. For example, if lb(2)=ub(2), fmincon gives this error: Equal upper and lower bounds not permitted in this large-scale method. Use equality constraints and the medium-scale method instead.

When using fmincon, the initial value/starting point we choose determines the minimum value that will result. There may be more than one minimum value for a given function, but only one will be the absolute min. We want to make sure we have the absolute minimum.

Tools in Matlab

Companion Matrix

In Matlab, a companion matrix of the monic polynomial $p(t) = c_0 + c_1 t + \ldots + c_{n-1}t^{n-1} + t^n$ is a square matrix with ones on the subdiagonal and a negative times each of the coefficients of the polynomial in the first row. The coefficients are listed from left to right in decreasing order, i.e., $c_{n-1}, c_{n-2}..., c_0$. A=compan(u) returns the corresponding companion matrix whose first row is $\frac{-u(2:n)}{u(1)}$, where u is of polynomial coefficients. The eigenvalues of compan(u) are the roots of the polynomial.

Example of Compan Command

The polynomial

$$(x-1)(x-2)(x-3) = x^3 - 7x + 6 \tag{.1}$$

results in $u = [1 \quad 0 \quad -7 \quad 6]$ and if A=compan(u), we get

$$A = \begin{pmatrix} 0 & 7 & -6 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

The eigenvalues are the polynomial roots. Therefore eig(compan(u))=3, 2, 1.

Kronecker Tensor Product

K=kron(X, Y) returns the Kronecker Tensor Product of X and Y. The result is a large array formed by taking all possible products between the elements of X and those of Y. If X is $m \times n$ and Y is $p \times q$, then kron(X, Y) is $mp \times nq$.

Example of Kronecker Product Command in Matlab

If X is 2×3,

$$kron(X,Y) = \begin{pmatrix} X(1,1)*Y & X(1,2)*Y & X(1,3)*Y \\ X(2,1)*Y & X(2,2)*Y & X(2,3)*Y \end{pmatrix}$$

Since our problem set up involves block matrices (which include the identity and zero matrix), the Kronecker product command will be very useful. The number of blocks in the block matrices is determined based on K, which we will choose to be any number between 3 and 20. As a result, when K is large, using the Kronecker product to construct the block matrices will come in handy.

Optimization Toolbox

The optimization toolbox consists of functions that perform maximization/minimization on linear/nonlinear constrained/non-constrained objective functions.

Usually these routines require that the objective functions be defined in M-files. Alternatively, a string variable containing a Matlab expression, with x representing the independent variables, can be used. Optional arguments can be used in the routines to change the optimization parameters and place bounds on the variables.

Replicating Matrices

Given a matrix A, B=repmat(A,m,n) creates a large matrix B consisting of an $m \times n$ tiling of copies of A. repmat(A,n) creates an $n \times n$ tiling.

Example of Repmat Command

B=repmat(eye(2),3,4) gives

$$B = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

For the above example, A=eye(2,2).

Reshape Command

B=reshape(A, m, n) returns the mxn matrix whose elements are taken column wise from A. You will get an error if A does not have mxn elements in it.

Example of the Reshape Command in Matlab

Let

$$A = \begin{pmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{pmatrix}$$

Then, if B=reshape(A,2,6),

$$B = \begin{pmatrix} 1 & 3 & 5 & 7 & 9 & 11 \\ 2 & 4 & 6 & 8 & 10 & 12 \end{pmatrix}$$

Size Command

Suppose A is an $m \times n$ matrix. v=size(A,1) gives you m. v=size(A,2) gives you n. So, the 1 or 2(or 3...) specifies if you want the length or width of the matrix or array. If a positive number other than 1 or 2 is entered, then Matlab outputs a default value of 1.

Matlab Code

```
%Chapter 2
%Section 2.1
%Example 1
%This routine computes the minimal proper test signal v*
%for Example 1.
%We determine v* by calculating the SVD of pinv(H)*X.
%Hs below is the pseudoinverse of H multiplied by X
%(which was determined in p1.m).

function v = p0(H0,H1,X)
    H=[H0, -H1];
    Hs=pinv(H)*X;
    [U,S,V]=svd(Hs); %This determines a svd of Hs.
    %This following gives the minimal proper test signal.
    v=(1/S(1,1))*V(:,1);
end

%Example 1 Data File
%This M-file defines the input variables that are used
%to compute the minimal test signal v* for Example 1.
%The following code will work for any K value greater than one.

function [H0,H1,X,m] = p1(K)

K=7;
%This K is a numerical value that indicates the number of time steps.
%If we change K here, we must change it to the same value in pd.m.

%Below is a set of matrices used for Example 1.
%To check to make sure the code is reasonable we try a
%value for K>1.

A0=[1 -1;1 1]; %System input matrix for model 0
A1=[1 -1;0 1]; %System input matrix for model 1
B0=[1 2;1 3]; %Input matrix for model 0
B1=B0; %Input matrix for model 1
C0=[0 1]; %System output matrix for model 0
C1=C0; %System output matrix for model 1
M0=[1 1 0;0 1 0]; %Additive uncertainty matrix for input model 0
```

```
M1=M0; %Additive uncertainty matrix for input model 1
N0=[0 0 1]; %Additive uncertainty matrix for output model 0
N1=N0; %Additive uncertainty matrix for output model 1

%Variable dimensions for the model.

[n,n]=size(A0); %This gives the dimensions of the matrix A0.
[n,n]=size(A1); %This gives the dimensions of the matrix A1.
[n,m]=size(B0); %This gives the dimensions of the matrix B0.
[n,m]=size(B1); %This gives the dimensions of the matrix B1.
[r,n]=size(C0); %This gives the dimensions of the matrix C0.
[r,n]=size(C1); %This gives the dimensions of the matrix C1.
[n,p]=size(M0); %This gives the dimensions of the matrix M0.
[n,p]=size(M1); %This gives the dimensions of the matrix M1.
[r,p]=size(N0); %This gives the dimensions of the matrix N0.
[r,p]=size(N1); %This gives the dimensions of the matrix N1.

u=eye(1,K+1);
%u gives the missing elements needed to complete
%the companion matrix.
A=-compan(u);
%A is the companion matrix that has -1 on the subdiagonal
%and zeros elsewhere.
%A is then multiplied by A_i using the kronecker command.

%This forms the 1st column of the M_0 matrix.
m0=[A0; repmat(zeros(n,n), K-1,1)];
%This forms the 1st column of the M_1 matrix.
m1=[A1; repmat(zeros(n,n), K-1,1)];
%This forms the last column of each M_i matrix.
m2=[repmat(zeros(n,p),K,1)];

%This forms the 1st column of the N_0 matrix.
n0=[C0; repmat(zeros(r,n),K,1)];
%This forms the 1st column of the N_1 matrix.
n1=[C1; repmat(zeros(r,n),K,1)];

%These are block matrices whose block size is determined by K.
%Therefore, the block size for the matrices
%will vary depending K.
```

```
E0=eye(K*n,K*n)+kron(A,A0);
E1=eye(K*n,K*n)+kron(A,A1);
M_0=[m0, kron(eye(K,K),M0), m2];
M_1=[m1, kron(eye(K,K),M1), m2];
B_0=kron(eye(K,K),B0);
B_1=kron(eye(K,K),B1);
C_0=[repmat(zeros(r,n),1,K); kron(eye(K,K),C0)];
C_1=[repmat(zeros(r,n),1,K); kron(eye(K,K),C1)];
N_0=[n0, kron(eye((K+1),(K+1)),N0)];
N_1=[n1, kron(eye((K+1),(K+1)),N0)];

%The matrices below are used to form the two y^i models.
%That is, y^i = X_i*v + H_i*u^i.

H0=C_0*(inv(E0))*M_0+N_0;
%H0 is multiplied by the additive uncertainty term in y^0.
H1=C_1*(inv(E1))*M_1+N_1;
%H1 is multiplied by the additive uncertainty term in y^1.
%This matrix is multiplied by the test signal in y^0.
X0=C_0*(inv(E0))*B_0;
%This matrix is multiplied by the test signal in y^1.
X1=C_1*(inv(E1))*B_1;
X=X1-X0; %X is used in p0.m.

%Example 1 Program Driver
%This is a driver file which accepts k as the input.
%This routine creates a plot of v as a function of k.

clear
figure(1)
clf

K=7; %This value is defined in p1.m.
%We choose K as any number between [3,20].
%K is not a global variable, so it must be manually
%changed in p1.m and pd.m.

[H0,H1,X,m] = p1(K)
%This calls the program that defines the
%block matrices based on K.  It determines X_i and H_i.
```

```
v = p0(H0,H1,X)
%This calls the program that calculates
%the minimum test signal, v*.  v* is a long vector Kn by 1.
vd = reshape(v,m,K)
%This changes v* into an n by K matrix.
%Each column represents the value of v* at each time step.
%For instance, the first column of the matrix corresponds to k=1.
%The second column corresponds to k=2, etc.

%This plots v* as a function of k.

vectM = [1 2 3 4 5];
colors = ['-r' '-b' '-g' '-k' '-m'];

for i=1:m
    M = vectM(i);
    k=0:K-1;
    figure(1)
    hold on
    plot(k,vd(i,:),colors(2*i-1:2*i))
%This graphs the first component of v over K time steps.
%Since m=2, v=[v0;v1].
    axis 'auto'
    xlabel('k');
    ylabel('v');
    title('Plot of v vs. k')
    legend('v_0','v_1');
    hold off
end

%Section 2.2
%Example 2
%This routine solves the incipient problem with one
%fault for the optimal test signal w*, which is used
%to determine v*_inc = (1/dt)w*.

function w = t0(Ht,Xtp)
    H=[Ht, -Ht]; %Ht represents H(\theta).
    Hp=pinv(H)*Xtp;
%This is the pseudoinverse of H multiplied by Xtp.
%Xtp represents X'(\theta).
```

```
    [U,S,V]=svd(Hp); %This determines a svd of Hp.
    %Due to approximation error this gives at least a
    %suboptimal proper test signal.
    w=V(:,1)/S(1,1);
end

%Example 2
%This routine solves for v2* in the incipient problem
%with one parameter.
%This is done for the faulty case using the two model method.
%H2 is the pseudoinverse of H multiplied by Xtd.

function v2 = t02(Ht,Xtd)
    H=[Ht, -Ht]; %Ht represents H(\theta).
    H2=pinv(H)*Xtd;
    [U,S,V]=svd(H2); %This determines a svd of H2.
    %This is v* for the faulty incipient
    %one parameter problem.
    v2=V(:,1)/S(1,1);
end

%Example 2 Data File
%This M-file defines the input variables that are
%used to compute the %minimum control w*,
%which is computed in t0.m.

function [Xt,Xtp,Ht,Xtd] = t1

t=0;  %t represents \theta, which will be a fixed value.
dt=0.1; %This dt represents \delta\theta, a small error term.
%dt must be the same as the dt value in td.m.
%It will be either 0.1, 0.5, 1.0, or 10.0.
%Whatever value is chosen in td.m, must be the same value here.

Xt=[1+t t^2; 2+(3*t) 2*(t^2); t^2+t 1];
%Xt is X(\theta) which will be given.

Xtp=[1 2*t; 3 4*t; 2*t+1 0];
%Xtp is the derivative of X(\theta) with respect to \theta.

Ht=[1 0 0;0 1 0;0 0 1];
```

```
%Ht is given and it represents H(\theta).
%This particular choice for H(\theta) is a constant matrix.

Xtd=[dt (2*t*dt)+(dt)^2; 3*(dt) (4*t*dt)+(2*(dt)^2);
2*t*dt+((dt)^2)+dt 0];
%Xtd is the difference between X(\theta+\delta\theta), X(\theta).

return

%Example 2 Driver
%This is a driver file that plots v* and v*_inc on the same graph.
%This routine creates a plot of v* and v*_inc as a function of time.

clear
figure(1)
clf

dt=0.1;
%dt represents \delta\theta, a small error term.
%We choose either 0.1, 0.5, 1.0, or 10.0 and
%it is the same value as the dt in t1.m.

%Given t and dt, this program calculates Xt, Xtp, Ht, Xtd.
[Xt,Xtp,Ht,Xtd] = t1;

w = t0(Ht,Xtp)
%This calls the program that calculates the
%minimum test signal w*.  w* is used to determine v*_inc.
vi = (1/dt)*w  %This computes v*_{inc}.
v2 = t02(Ht,Xtd)
%This calls the program that calculates the test signal v*.
%We use the code from the two model static case
%with appropriate modifications.

%This plot compares v* and v*_inc for specified values
%of dt and t=0.

figure(1)
hold on
plot(v2,'-r')
plot(vi,'-b')
```

```
axis 'auto'
xlabel('k');
ylabel('Test signal');
title('Plot Comparing v* and v*_{inc}')
legend('v*','v*_{inc}');
hold off


%Chapter 3
%Section 3.1
%Example 4
%Safe Process Paper (SPP) Second Example
%This function represents the matrix X(\psi).

function X2=xfn(t,p)
X2=[1+t t^2; 2+(3*p)-(2*t) 2*(p^2); (t*p)+1 1];
%This matrix is X, which is given in the example.
return

%Example 4
%SPP Example 2
%This function is the derivative of X(\psi)
%with respect to \theta.

function Xt2=xdtfn(t,p)
Xt2=[1 2*t; -2 0; p 0];
%This is the derivative of the given matrix X
%with respect to \theta.
return

%Example 4
%SPP Example 2
%This function is the derivative of X(\psi)
%with respect to \phi.

function Xp2=xdpfn(t,p)
Xp2=[0 0; 3 4*p; t 0];
%This is the derivative of the given matrix X
%with respect to \phi.
return
```

```
%The function below solves for the standard
%test signal v in Ex.2 in SPP.
%The method is used for the one parameter
%incipient problem and the static case.

%The standard test signal depends on E and dk
%and we take 0 \leq E \leq 1.
function v = vstar(E,dk)
dt=E*dk;
dp=(1-E)*dk;
x1=xfn(0,0); %Normal model
x2=xfn(dt,dp); %Faulty model
H3=[eye(3,3),-eye(3,3)];
[U,S,V]=svd(pinv(H3)*(x1-x2));
sig = S(1,1); %Largest singular value of pinv(H3)*(x1-x2)
gam = V(:,1); %Right singular vector that corresponds to sig.
%Test signal for special case found using the two model method.
v = (1/sig)*gam;
return

%Example 4
%SPP Example 2
%This function evaluates the inner min of the
%optimization function over E.
%Given some x, we call this function in
%graphin.m and assign a value to x.

function fval = innermin(x)

E0 = 0.25; %This is an initial guess for the minimum E.

w=[cos(x);sin(x)]; %norm(w)=1 is a constraint
Hd=pinv([eye(3,3),-eye(3,3)]);
a=Hd*xdtfn(0,0); %Substitution for the optmization function.
b=Hd*xdpfn(0,0); %Substitution for the optmization function.
inmaxval=@(E)(norm(E*a*w + (1-E)*b*w))^2; %Optimization function

options=optimset('LargeScale','off');
[EE, fval, exitflag, output] =
fmincon(inmaxval,E0,[],[],[],[],0,1,[],options)
```

```
return

%Example 4
%SPP Example 2
%The loop below creates a plot of innermin(x)
%for different values of x.

x=0:0.1:2*pi;
for i=1:length(x)
fval(i) = innermin(x(i));
end
plot(x,fval)

%Example 4
%SPP Example 2
%This is the driver file.

%This command clears the plot each time this M-file is run.

clear
figure(1)
clf

%This graphs the standard test signals and the
%incipient test signal.
%We plot the test signals for different values of dk.

fval = innermin(0);
dk=.1; % This is the small error term and it will vary.

%Incipient test signal: v_inc=(1/dk)*(1/\zeta)*w.

vinc = (1/dk)*(1/sqrt(abs(fval)))*[cos(0);sin(0)]
%v_i=vstar(E,dk) are test signals found using
%the two model method.
v1=vstar(1,dk);
v03=vstar(0.3,dk);
v05=vstar(0.5,dk);
v06=vstar(0.6,dk);
v07=vstar(0.7,dk);
v08=vstar(0.8,dk);
```

```
v09=vstar(0.9,dk);
v0=vstar(0,dk);

%Figures 3.2, 3.3, and 3.4 in Chapter 3 are
%examples from the Safe Process Paper.
%We plot v and v_inc for dk=0.1,1 and
%E=0, 0.3, 0.5, 0.7, 0.9, 1.

x_val1 = [0 v1(1)]';
y_val1 = [0 v1(2)]';
axis([0 25 -5 25])
hold on
plot(x_val1,y_val1,'-k')
hold on
x_val2 = [0 v03(1)]';
y_val2 = [0 v03(2)]';
plot(x_val2,y_val2,'-k')
hold on
x_val3 = [0 v05(1)]';
y_val3 = [0 v05(2)]';
plot(x_val3,y_val3,'-k')
hold on
x_val4 = [0 v06(1)]';
y_val4 = [0 v06(2)]';
plot(x_val4,y_val4,'-k')
hold on
x_val5 = [0 v07(1)]';
y_val5 = [0 v07(2)]';
plot(x_val5,y_val5,'-k')
hold on
x_val6 = [0 v08(1)]';
y_val6 = [0 v08(2)]';
plot(x_val6,y_val6,'-k')
hold on
x_val7 = [0 v09(1)]';
y_val7 = [0 v09(2)]';
plot(x_val7,y_val7,'-k')
hold on
x_val8 = [0 vinc(1)]';
y_val8 = [0 vinc(2)]';
plot(x_val8,y_val8,'-r')
```

```
hold on

%Section 3.2
%San Antonio Paper Example 2
%This routine computes the minimal proper
%standard test signal v*.
%v* is found by calculating the SVD of
%pinv(H)*X, which are calculated in inc1.m.

function v = inc0(H0,H1,X)
H=[H0, -H1];
[U,S,V]=svd(pinv(H)*X);
sig = S(1,1);
gam = V(:,1);
v = (1/sig)*gam;
return

%San Antonio Paper Example 2
%This M-file defines the input variables
%that will be used to compute the minimal test signal v*.

%The following code should work for any K value
%greater than one.
%This file corresponds with inc0.m,
%which finds the standard test signals.

function [H0,H1,X] = inc1(K,dm,dk,dt)

K=5;
%This K is a numerical value that indicates
%the number of time steps.
%K must be the same value in inc11.m, inc12.m, inc13.m, incd.m.

%Below is a set of matrices we will use for
%the test problem to check to see if the code is reasonable.
t1=2;
k=12;
d=0.2;
h=0.2;
%This step size h may vary and must be the same in
%inc11.m, inc12.m, inc13.m.
```

```
%dm,dk,dt may vary so they are defined in the function
%which calls this input file.

%System input matrix for model 0
A0=[1 h;-h*k*t1 -h*t1*d];
%System input matrix for model 1
A1=[1 h;-h*(k+dk)*(t1+dm) -h*(t1+dm)*(d+dt)];
B0=[0; h*t1]; %Input matrix for model 0
B1=B0; %Input matrix for model 1
C0=[0 1]; %System output matrix for model 0
C1=C0; %System output matrix for model 1
%Additive uncertainty matrix for input model 0
M0=[1 1 0;0 1 0];
%Additive uncertainty matrix for input model 1
M1=M0;
%Additive uncertainty matrix for output model 0
N0=[0 0 1];
%Additive uncertainty matrix for output model 1
N1=N0;

%Variable dimensions for the model.

[n,n]=size(A0); %This gives the dimensions of the matrix A0.
[n,n]=size(A1); %This gives the dimensions of the matrix A1.
[n,m]=size(B0); %This gives the dimensions of the matrix B0.
[n,m]=size(B1); %This gives the dimensions of the matrix B1.
[r,n]=size(C0); %This gives the dimensions of the matrix C0.
[r,n]=size(C1); %This gives the dimensions of the matrix C1.
[n,p]=size(M0); %This gives the dimensions of the matrix M0.
[n,p]=size(M1); %This gives the dimensions of the matrix M1.
[r,p]=size(N0); %This gives the dimensions of the matrix N0.
[r,p]=size(N1); %This gives the dimensions of the matrix N1.

u=eye(1,K+1);
%u gives the missing elements needed to
%complete the companion matrix,
A=-compan(u);
%A is the companion matrix that has -1 on
%the subdiagonal and zeros elsewhere.
%A is then multiplied by A_i using the kronecker command.
```

```
%This forms the 1st column of the M_0 matrix.
m0=[A0; repmat(zeros(n,n), K-1,1)];
%This forms the 1st column of the M_1 matrix.
m1=[A1; repmat(zeros(n,n), K-1,1)];
%This forms the last column of each M_i matrix.
m2=[repmat(zeros(n,p),K,1)];

%This forms the 1st column of the N_0 matrix.
n0=[C0; repmat(zeros(r,n),K,1)];
%This forms the 1st column of the N_1 matrix.
n1=[C1; repmat(zeros(r,n),K,1)];

%These are block matrices whose block size is
%determined by K.  Therefore, the block size for
%the matrices will vary depending on K.

E0=eye(K*n,K*n)+kron(A,A0);
E1=eye(K*n,K*n)+kron(A,A1);
M_0=[m0, kron(eye(K,K),M0), m2];
M_1=[m1, kron(eye(K,K),M1), m2];
B_0=kron(eye(K,K),B0);
B_1=kron(eye(K,K),B1);
C_0=[repmat(zeros(r,n),1,K); kron(eye(K,K),C0)];
C_1=[repmat(zeros(r,n),1,K); kron(eye(K,K),C1)];
N_0=[n0, kron(eye((K+1),(K+1)),N0)];
N_1=[n1, kron(eye((K+1),(K+1)),N0)];

%The matrices below are used to form the two y^i
%models, i.e., y^i = X_i*v + H_i*u^i.

H0=C_0*(inv(E0))*M_0+N_0;
%H0 is multiplied by the additive uncertainty term in y^0.
H1=C_1*(inv(E1))*M_1+N_1;
%H1 is multiplied by the additive uncertainty term in y^1.
%This matrix is multiplied by the test signal in y^0.
X0=C_0*(inv(E0))*B_0;
%This matrix is multiplied by the test signal in y^1.
X1=C_1*(inv(E1))*B_1;
X=X0-X1; %X is one of the terms used in inc0.m.
```

```
%San Antonio Paper Example 2
%This M-file defines the input variables that
%will be used to compute the minimal test signal v*.

%This is the input file for xt1.m and it treats
%\theta_1=m=t1 as a variable.
%This file corresponds with xt1.m.

function [C_0,C_1,E0,E1,B_0,B_1] = inc11(K,dm,dk,dt)

syms t1
%This input file leaves the variables symbolic.
%We find the partial derivative with respect to t1.

k=12;
d=0.2;

%dm,dk,dt may vary so they are defined in the function
%which calls this input file.

h=0.2; %This step size may vary.
%K is a numerical value that indicates the number
%of time steps.
K=5;
%If we change K here, we must also change it to
%the same value in incd.m.

%Below is a set of matrices we will use for the
%test problem to check to see if the code is reasonable.

%System input matrix for model 0
A0=[1 h;-h*k*t1 -h*t1*d];
%System input matrix for model 1
A1=[1 h;-h*(k+dk)*(t1+dm) -h*(t1+dm)*(d+dt)];
B0=[0; h*t1]; %Input matrix for model 0
B1=B0; %Input matrix for model 1
C0=[0 1]; %System output matrix for model 0
C1=C0; %System output matrix for model 1
M0=[1 1 0;0 1 0]; %Additive uncertainty matrix for input model 0
M1=M0; %Additive uncertainty matrix for input model 1
N0=[0 0 1]; %Additive uncertainty matrix for output model 0
```

```
N1=N0; %Additive uncertainty matrix for output model 1

%Variable dimensions for the model.

[n,n]=size(A0); %This gives the dimensions of the matrix A0.
[n,n]=size(A1); %This gives the dimensions of the matrix A1.
[n,m]=size(B0); %This gives the dimensions of the matrix B0.
[n,m]=size(B1); %This gives the dimensions of the matrix B1.
[r,n]=size(C0); %This gives the dimensions of the matrix C0.
[r,n]=size(C1); %This gives the dimensions of the matrix C1.
[n,p]=size(M0); %This gives the dimensions of the matrix M0.
[n,p]=size(M1); %This gives the dimensions of the matrix M1.
[r,p]=size(N0); %This gives the dimensions of the matrix N0.
[r,p]=size(N1); %This gives the dimensions of the matrix N1.

u=eye(1,K+1);
%u gives the missing elements needed to complete
%the companion matrix.
A=-compan(u);
%A creates the companion matrix that has -1 on the
%subdiagonal and zeros elsewhere.
%A is then multiplied by A_i using the kronecker command.

%This forms the 1st column of the M_0 matrix.
m0=[A0; repmat(zeros(n,n), K-1,1)];
%This forms the 1st column of the M_1 matrix.
m1=[A1; repmat(zeros(n,n), K-1,1)];
%This forms the last column of each M_i matrix.
m2=[repmat(zeros(n,p),K,1)];

%This forms the 1st column of the N_0 matrix.
n0=[C0; repmat(zeros(r,n),K,1)];
%This forms the 1st column of the N_1 matrix.
n1=[C1; repmat(zeros(r,n),K,1)];

%These are block matrices whose block size is
%determined by K.  Therefore, the block size for
%the matrices will vary depending on K.

E0=eye(K*n,K*n)+kron(A,A0);
E1=eye(K*n,K*n)+kron(A,A1);
```

```
M_0=[m0, kron(eye(K,K),M0), m2];
M_1=[m1, kron(eye(K,K),M1), m2];
B_0=kron(eye(K,K),B0);
B_1=kron(eye(K,K),B1);
C_0=[repmat(zeros(r,n),1,K); kron(eye(K,K),C0)];
C_1=[repmat(zeros(r,n),1,K); kron(eye(K,K),C1)];
N_0=[n0, kron(eye((K+1),(K+1)),N0)];
N_1=[n1, kron(eye((K+1),(K+1)),N0)];

%The matrices below are used to form the two y^i models.
%That is, y^i = X_i*v + H_i*u^i.

H0=C_0*(inv(E0))*M_0+N_0;
%H0 is multiplied by the additive uncertainty term in y^0.
H1=C_1*(inv(E1))*M_1+N_1;
%H1 is multiplied by the additive uncertainty term in y^1.
%This matrix is multiplied by the test signal in y^0.
X0=C_0*(inv(E0))*B_0;
%This matrix is multiplied by the test signal in y^1.
X1=C_1*(inv(E1))*B_1;
X=X1-X0; %X is used in inc0.m.

%San Antonio Paper Example 2
%This M-file defines the input variables that will be used
%to compute the minimal test signal v*.

%This is the input file for xt2.m and it treats
%\theta_2=k as a variable.  This file corresponds with xt2.m.

function [C_0,C_1,E0,E1,B_0,B_1] = inc12(K,dm,dk,dt)

syms k
%This input file leaves the variables symbolic.

t1=2;
d=0.2;

%dm,dk,dt may vary so they are defined in the function
%which calls this input file.

h=0.2; %This step size may vary.
```

```
%This K is a numerical value that indicates the
%number of time steps.
K=5;
%If we change K here, we must change it to the
%same value in incd.m.

%Below is a set of matrices we will use for the
%test problem to check to see if the code is reasonable.

%System input matrix for model 0
A0=[1 h;-h*k*t1 -h*t1*d];
%System input matrix for model 1
A1=[1 h;-h*(k+dk)*(t1+dm) -h*(t1+dm)*(d+dt)];
B0=[0; h*t1]; %Input matrix for model 0
B1=B0; %Input matrix for model 1
C0=[0 1]; %System output matrix for model 0
C1=C0; %System output matrix for model 1
M0=[1 1 0;0 1 0]; %Additive uncertainty matrix for input model 0
M1=M0; %Additive uncertainty matrix for input model 1
N0=[0 0 1]; %Additive uncertainty matrix for output model 0
N1=N0; %Additive uncertainty matrix for output model 1

%Variable dimensions for the model.

[n,n]=size(A0); %This gives the dimensions of the matrix A0.
[n,n]=size(A1); %This gives the dimensions of the matrix A1.
[n,m]=size(B0); %This gives the dimensions of the matrix B0.
[n,m]=size(B1); %This gives the dimensions of the matrix B1.
[r,n]=size(C0); %This gives the dimensions of the matrix C0.
[r,n]=size(C1); %This gives the dimensions of the matrix C1.
[n,p]=size(M0); %This gives the dimensions of the matrix M0.
[n,p]=size(M1); %This gives the dimensions of the matrix M1.
[r,p]=size(N0); %This gives the dimensions of the matrix N0.
[r,p]=size(N1); %This gives the dimensions of the matrix N1.

u=eye(1,K+1);
%u gives the missing elements needed to
%complete the companion matrix.
A=-compan(u);
%A is the companion matrix that has -1 on the
%subdiagonal and zeros elsewhere.
```

```
%A is then multiplied by A_i using the kronecker command.

%This forms the 1st column of the M_0 matrix.
m0=[A0; repmat(zeros(n,n), K-1,1)];
%This forms the 1st column of the M_1 matrix.
m1=[A1; repmat(zeros(n,n), K-1,1)];
%This forms the last column of each M_i matrix.
m2=[repmat(zeros(n,p),K,1)];

%This forms the 1st column of the N_0 matrix.
n0=[C0; repmat(zeros(r,n),K,1)];
%This forms the 1st column of the N_1 matrix.
n1=[C1; repmat(zeros(r,n),K,1)];

%These are block matrices whose block size is
%determined by K.  Therefore, the block size for the
%matrices will vary depending on our choice of K.

E0=eye(K*n,K*n)+kron(A,A0);
E1=eye(K*n,K*n)+kron(A,A1);
M_0=[m0, kron(eye(K,K),M0), m2];
M_1=[m1, kron(eye(K,K),M1), m2];
B_0=kron(eye(K,K),B0);
B_1=kron(eye(K,K),B1);
C_0=[repmat(zeros(r,n),1,K); kron(eye(K,K),C0)];
C_1=[repmat(zeros(r,n),1,K); kron(eye(K,K),C1)];
N_0=[n0, kron(eye((K+1),(K+1)),N0)];
N_1=[n1, kron(eye((K+1),(K+1)),N0)];

%The matrices below are used to form the two y^i models.
%That is, y^i = X_i*v + H_i*u^i.

H0=C_0*(inv(E0))*M_0+N_0;
%H0 is multiplied by the additive uncertainty term in y^0.
H1=C_1*(inv(E1))*M_1+N_1;
%H1 is multiplied by the additive uncertainty term in y^1.
%This matrix is multiplied by the test signal in y^0.
X0=C_0*(inv(E0))*B_0;
%This matrix is multiplied by the test signal in y^1.
X1=C_1*(inv(E1))*B_1;
X=X1-X0; %X is used in inc0.m.
```

```
%San Antonio Paper Example 2
%This M-file defines the input variables that will
%be used to compute the minimal test signal v*.

%This is the input file for xt3.m and it treats
%\theta_3=d as a variable.
%This file corresponds with xt3.m.

function [C_0,C_1,E0,E1,B_0,B_1] = inc13(K,dm,dk,dt)

syms d
%This input file leaves the variables symbolic.

t1=2;
k=12;

%dm,dk,dt may vary so they are defined in the function
%which calls this input file.

h=0.2; %This step size may vary.
K=5; %This K is a numerical value that indicates the
%number of time steps.
%If we change K here, we must change it to the
%same value in incd.m.

%Below is a set of matrices we will use for the
%test problem to check to see if the code is reasonable.

%System input matrix for model 0
A0=[1 h;-h*k*t1 -h*t1*d];
%System input matrix for model 1
A1=[1 h;-h*(k+dk)*(t1+dm) -h*(t1+dm)*(d+dt)];
B0=[0; h*t1]; %Input matrix for model 0
B1=B0; %Input matrix for model 1
C0=[0 1]; %System output matrix for model 0
C1=C0; %System output matrix for model 1
M0=[1 1 0;0 1 0]; %Additive uncertainty matrix for input model 0
M1=M0; %Additive uncertainty matrix for input model 1
N0=[0 0 1]; %Additive uncertainty matrix for output model 0
N1=N0; %Additive uncertainty matrix for output model 1
```

```
%Variable dimensions for the model.

[n,n]=size(A0); %This gives the dimensions of the matrix A0.
[n,n]=size(A1); %This gives the dimensions of the matrix A1.
[n,m]=size(B0); %This gives the dimensions of the matrix B0.
[n,m]=size(B1); %This gives the dimensions of the matrix B1.
[r,n]=size(C0); %This gives the dimensions of the matrix C0.
[r,n]=size(C1); %This gives the dimensions of the matrix C1.
[n,p]=size(M0); %This gives the dimensions of the matrix M0.
[n,p]=size(M1); %This gives the dimensions of the matrix M1.
[r,p]=size(N0); %This gives the dimensions of the matrix N0.
[r,p]=size(N1); %This gives the dimensions of the matrix N1.

u=eye(1,K+1);
%u gives the missing elements needed to
%complete the companion matrix.
A=-compan(u);
%A is the companion matrix that has -1 on the
%subdiagonal and zeros elsewhere.
%A is then multiplied by A_i using the kronecker command.

%This forms the 1st column of the M_0 matrix.
m0=[A0; repmat(zeros(n,n), K-1,1)];
%This forms the 1st column of the M_1 matrix.
m1=[A1; repmat(zeros(n,n), K-1,1)];
%This forms the last column of each M_i matrix.
m2=[repmat(zeros(n,p),K,1)];

%This forms the 1st column of the N_0 matrix.
n0=[C0; repmat(zeros(r,n),K,1)];
%This forms the 1st column of the N_1 matrix.
n1=[C1; repmat(zeros(r,n),K,1)];

%These are block matrices whose block size is
%determined by K.  Therefore, the block size for
%the matrices will vary depending on our choice of K.

E0=eye(K*n,K*n)+kron(A,A0);
E1=eye(K*n,K*n)+kron(A,A1);
M_0=[m0, kron(eye(K,K),M0), m2];
```

```
M_1=[m1, kron(eye(K,K),M1), m2];
B_0=kron(eye(K,K),B0);
B_1=kron(eye(K,K),B1);
C_0=[repmat(zeros(r,n),1,K); kron(eye(K,K),C0)];
C_1=[repmat(zeros(r,n),1,K); kron(eye(K,K),C1)];
N_0=[n0, kron(eye((K+1),(K+1)),N0)];
N_1=[n1, kron(eye((K+1),(K+1)),N0)];

%The matrices below are used to form the two y^i models.
%That is, y^i = X_i*v + H_i*u^i.

H0=C_0*(inv(E0))*M_0+N_0;
%H0 is multiplied by the additive uncertainty term in y^0.
H1=C_1*(inv(E1))*M_1+N_1;
%H1 is multiplied by the additive uncertainty term in y^1.
%This matrix is multiplied by the test signal in y^0.
X0=C_0*(inv(E0))*B_0;
%This matrix is multiplied by the test signal in y^1.
X1=C_1*(inv(E1))*B_1;
X=X1-X0; %X is one of the terms used in inc0.m.

%San Antonio Paper Example 2
%This function is the derivative of X(\theta)
%with respect to \theta1=m.
%The formula from Section 3 of the
%Safe Process Paper is used.
%The partial has been calculated.
%Then a numerical value is assigned to m=t1.

function Xmd=xt1(t1,k,d)

syms t1 %Treats t1 as a variable.

%%[C_0,C_1,E0,E1,B_0,B_1] = inc11(K,dm,dk,dt);
[C_0,C_1,E0,E1,B_0,B_1] = inc11(5,-1.1,-4,0.2);
%Here we take K=5.
%Choose values for the drift parameters,
%which will vary depending on the fault condition.

%Be sure to use the same values for K and the drift parameters
%in all of the programs related to this problem,
%except plotin2.m, out2.m, inc0.m.
%We find the partial derivative with respect to each variable
%so that we can determine our optimization function.

C_0pd=@(t1) diff(C_0,t1);

B_0pd=@(t1) diff(B_0,t1);

E0pd=@(t1) diff(E0,t1);

C_1pd=@(t1) diff(C_1,t1);

B_1pd=@(t1) diff(B_1,t1);

E1pd=@(t1) diff(E1,t1);

X_m0=(C_0pd(t1))*inv(E0)*(B_0)-(C_0)*inv(E0)*(E0pd(t1))*inv(E0)*
(B_0+(C_0)*inv(E0)*(B_0pd(t1));

X_m1=(C_1pd(t1))*inv(E1)*(B_1)-(C_1)*inv(E1)*(E1pd(t1))*inv(E1)*
(B_1)+(C_1)*inv(E1)*(B_1pd(t1));

Xm= X_m1-X_m0;

Xmd = subs(Xm, t1, 2);

return

%San Antonio Paper Example 2
%This function is the derivative the matrix X(\theta)
%with respect to \theta2=k.
%The formula from Section 3 of the
%Safe Process Paper is used.
%After the partial has been calculated.
%Then a numerical value is assigned to k.

function Xkd=xt2(t1,k,d)

syms k %Treats k as a variable.

%%[C_0,C_1,E0,E1,B_0,B_1] = inc12(K,dm,dk,dt);
```

```
[C_0,C_1,E0,E1,B_0,B_1] = inc12(5,-1.1,-4,0.2);
%Here we take K=5.
%Choose values for the drift parameters,
%which may vary within their given ranges.

%We must make sure we use the same values for K and
%the drift parameters in all of the programs related to
%this problem, except for plotin2.m, out2.m, inc0.m.

C_0pd=@(k) diff(C_0,k);

B_0pd=@(k) diff(B_0,k);

E0pd=@(k) diff(E0,k);

C_1pd=@(k) diff(C_1,k);

B_1pd=@(k) diff(B_1,k);

E1pd=@(k) diff(E1,k);

X_k0=(C_0pd(k))*inv(E0)*(B_0)-(C_0)*inv(E0)*(E0pd(k))*inv(E0)*
(B_0)+(C_0)*inv(E0)*(B_0pd(k));

X_k1=(C_1pd(k))*inv(E1)*(B_1)-(C_1)*inv(E1)*(E1pd(k))*inv(E1)*
(B_1)+(C_1)*inv(E1)*(B_1pd(k));

Xk= X_k1-X_k0;

Xkd = subs(Xk, k, 12);

return

%San Antonio Paper Example 2
%This is the derivative of matrix X(\theta)
%with respect to \theta3=d.
%The formula from Section 3 of the
%Safe Process Paper is used.
%After the partial has been calculated.
%Then a numerical value is assigned to d.
```

```
function Xdd=xt3(t1,k,d)

syms d %Treats d as a variable.

%%[C_0,C_1,E0,E1,B_0,B_1] = inc13(K,dm,dk,dt);
[C_0,C_1,E0,E1,B_0,B_1] = inc13(5,-1.1,-4,0.2);
%Here we take K=5.
%Choose values for the drift parameters,
%which may vary within their given ranges.

%Be sure to use the same values for K and the
%drift parameters in all of the programs related to
%this problem, except for plotin2.m, out2.m, inc0.m.

C_0pd=@(d) diff(C_0,d);

B_0pd=@(d) diff(B_0,d);

E0pd=@(d) diff(E0,d);

C_1pd=@(d) diff(C_1,d);

B_1pd=@(d) diff(B_1,d);

E1pd=@(d) diff(E1,d);

X_d0=(C_0pd(d))*inv(E0)*(B_0)-(C_0)*inv(E0)*(E0pd(d))*inv(E0)*
(B_0)+(C_0)*inv(E0)*(B_0pd(d));

X_d1=(C_1pd(d))*inv(E1)*(B_1)-(C_1)*inv(E1)*(E1pd(d))*inv(E1)*
(B_1)+(C_1)*inv(E1)*(B_1pd(d));

Xd=X_d1-X_d0;

Xdd = subs(Xd, d, 0.2);

return

%San Antonio Paper Example 2
%This function evaluates the inner min of the
%optimization function.
```

```
%This is done over the \alpha_i's given some x.

function z = innermin2(x)

K=5; %This is the final time step and will vary.

%This is an initial guess for the minimum \alpha1,
%\alpha2,\alpha3.
y0 = [-1.1; -4; 0.2];

%We want the norm(w)=1.
%Also, the row dimension of w must be the size of K,
%i.e. w is a Kx1 vector.
w=[sin(x(1))*sin(x(2))*cos(x(3));
sin(x(1))*sin(x(2))*sin(x(3))*cos(x(4));
sin(x(1))*sin(x(2))*sin(x(3))*sin(x(4));
cos(x(1));sin(x(1))*cos(x(2))];

%[H0,H1,X] = inc1(K,dm,dk,dt); %Calls input function file.
[H0,H1,X] = inc1(K,-1.1,-4,0.2);
%Input file with assigned values for the parameters
%and the drift parameters.
Hd=pinv([H0, -H1]);%The pseudoinverse of H is calculated here.

t1=2; %For the normal case, t1=m.
k=12; %Normal value for this mechanical system where k=s.
d=0.2; %Normal value for this mechanical system.

a=Hd*xt1(t1,k,d); %Substitution for the optmization function.
b=Hd*xt2(t1,k,d); %Substitution for the optmization function.
c=Hd*xt3(t1,k,d); %Substitution for the optmization function.

%Optimization function
inmaxval1=@(y)(100*(norm(y(1)*a*w + y(2)*b*w + y(3)*c*w))^2);

%options=optimset('LargeScale','off');
[yy, fval, exitflag, output] =
fmincon(inmaxval1,y0,[],[],[],[],[-1.1; -4; 0.2],
[-0.9; -2; 0.4]),[],options)

z=fval/100;
```

```
return

%San Anotnio Paper Example 2
%Optimization function used to find the max x

function z = myinner(x)
z = -innermin2(x);

%San Antonio Paper Example 2
%This function evaluates the outer max of the
%optimization function over x.

function vinc = outer2(z)

%This is an initial guess for the minimum a1,a2,a3.
x0 = [pi/4;pi/2;pi/4;pi/2];
a=z;  %Does nothing.
K=5;
h=0.2; %This step size may vary.

%Note that a wider interval is being used so that
%we do not get stuck at the edges as a solution.

myinner=@(x)(-innermin2(x));
options=optimset('LargeScale','off');
[xx, ffval, exitflag, output] =
fmincon(myinner,x0,[],[],[],[],[-1.1*pi;-1.2*pi;
-1.1*pi;1.1*pi],[1.1*pi;2.1*pi;1.1*pi;1.1*pi],[],options)

%This is the maximum value of the function.
ffval=-ffval

%We want to use the x that maximizes the
%optimization function.
x=xx;

%w is a vector and the row dimension of w will be
%the same as the value of K.
w=[sin(x(1))*sin(x(2))*cos(x(3));
sin(x(1))*sin(x(2))*sin(x(3))*cos(x(4));
sin(x(1))*sin(x(2))*sin(x(3))*sin(x(4));
cos(x(1));sin(x(1))*cos(x(2))];

%To calculate the incipient test signal we use:
%v_inc=(1/dk)*(1/\zeta)*w.
%dk=1; Since dk=1, we take \alpha_i=\delta\theta_i.
vinc=(1/dk)(1/sqrt(ffval))*w;

return

% San Antonio Paper Example 2
% This graphs the standard test signals and the
%incipient test signal for different a1,a2,a3
%values for Ex.2 in San Antonio Paper.

%This command clears the plot each time this M-file is run.

clear
figure(1)
clf

h=0.2; %This is the step size.
K=5; %We plot this test signal as a function of time t.
%Final time: K=5.

%This calls the function file that calculates the
%standard test signal.
%v=inc0(\alpha1,\alpha2,\alpha3)=inc0(dt1,dt2,dt3)=inc0(dm,dk,dt).

%[H0,H1,X] = inc1(K,dm,dk,dt) %Calls input function file.
[H0,H1,X] = inc1(K,-1.1,-4,0.2);
v0 = inc0(H0,H1,X);
[H0,H1,X] = inc1(K,-1.1,-4,0.4);
v1=inc0(H0,H1,X);
[H0,H1,X] = inc1(K,-1.1,-2,0.2);
v2=inc0(H0,H1,X);
[H0,H1,X] = inc1(K,-1.1,-2,0.4);
v3=inc0(H0,H1,X);
[H0,H1,X] = inc1(K,-0.9,-4,0.2);
v4=inc0(H0,H1,X);
[H0,H1,X] = inc1(K,-0.9,-4,0.4);
```

```
v5=inc0(H0,H1,X);
[H0,H1,X] = inc1(K,-0.9,-2,0.2);
v6=inc0(H0,H1,X);
[H0,H1,X] = inc1(K,-0.9,-2,0.4);
v7=inc0(H0,H1,X);

%We compare plots of v and v_inc for dk=1 and use
%different \alpha_i values.

%Incipient test signal: v_inc=(1/dk)*(1/\zeta)*w.
%Calculated in outer2.m.
v_inc = [18.2136; -8.6901; -2.8236; 0.0485; 0.0000]

kk=0:K-1;
plot(kk,v0)
hold on
plot(kk,v1)
hold on
plot(kk,v2)
hold on
plot(kk,v3)
hold on
plot(kk,v4)
hold on
plot(kk,v5)
hold on
plot(kk,v6)
hold on
plot(kk,v7)
hold on
plot(kk,v_inc,'-r')
hold on

%Chapter 4
%Section 4.6
%Continuous Time Example 1
%Solves inner minimum of the system,
%which we call 'val1', using bvp4c.
%This gives the solution to the boundary value problem.
%Untransformed Problem in paper entitled
%"Active incipient fault
```

```
%detection in continuous time systems with multiple faults"

%This is the innermost min for n=1.
%function val1 = innermin1(dt)
%This is the innermost min for n>1.
function val1 = innermin1(d,dt)

%We take norm(c)=1 and use the following parameterization.
%c=[cos(d(1)); sin(d(1))]; %For n=2
%For v 3-dimensional.
% c=[cos(d(1))
%    sin(d(1))*cos(d(2))
%    sin(d(1))*sin(d(2))];

%We take norm(c)=1 and use the following parameterization.
%For v 5-dimensional.
c=[cos(d(1))
   sin(d(1))*cos(d(2))
   sin(d(1))*sin(d(2))*cos(d(3))
   sin(d(1))*sin(d(2))*sin(d(3))*cos(d(4))
   sin(d(1))*sin(d(2))*sin(d(3))*sin(d(4))];

L=4; %\omega=L
solinit = bvpinit(linspace(0,L,12),@bvp2init);
options = bvpset('Stats','on','RelTol',1e-7);

%The following line integrates the
%system of differential equations on the
%interval [0,\omega]
%subject to two point boundary conditions.
sol = bvp4c(@bvp2,@bvp2bc,solinit,options);

t=linspace(0,L,50);
y=deval(sol,t);

% figure(1)
% plot(t,y(1:4,:))
% xlabel('Time')
% ylabel('State Variable z')
% title('Solutions for z(0)')
% legend('x1','x2','xhat1','xhat2')
% figure(2)
% plot(t,y(9,:))
% xlabel('Time')
% ylabel('Value of \eta')
% title('Solutions for \eta')

function yprime = bvp2(x,y)
%The components of y correspond to the original variables
%as y(1)=z, y(2)=\lambda, y(3)=\eta.
t1=0.1; %\theta_1
t2=1.0; %\theta_2
% dt(i)=\delta\theta_i %Perturbations

A=[-(t1) t2 0 0; 1 -(t1) 0 0; 0 0 -(t1+dt(1)) t2+dt(2);
0 0 1 -(t1+dt(1))]; %\tilde{A}
B=[0;1;0;1]; %\tilde{B}
C=[1 0 -1 0]; %\tilde{C}
M=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1]; %\tilde{M}

z=y(1:4); %This accounts for the first four rows
%of the first column.
lam=y(5:8);
%eta=y(9,1);
%vval=v(x); %For n=1
vval=v(c,x); %For n>1

yprime = [A*z+B*vval-0.5*M*M'*lam
          -A'*lam-C'*C*z
          0.25*lam'*M*M'*lam+0.5*z'*C'*C*z];
end

function res = bvp2bc(ya,yb)
%The bvp is solved with RES=0.
%The time interval is [0 L] and the boundary conditions are:
%
%    \lambda(0)=-0.5z(0), \lambda(L)=0, \eta(0)=0.
%
%ya corresponds to the initial time,
%yb corresponds to the final time.
res = [ya(5:8,1)+0.5*ya(1:4,1)
       yb(5:8,1)
       ya(9,1)];
end

function yinit = bvp2init(x)
%This is the initial guess for the solutions:
%z, \lambda, r, and q:
yinit = [0.5 0 0.25 0.5 0.5 0 1 1 0]';
end

%function vval = v(x) %For n=1
function vval = v(c,x) %For n>1
L=4; %\omega=L
g=sqrt(2/L);
%This is the form of the test signal v
%and also the optimization function.
%vval=g*sin(pi*x/L); %n=1
%vval=c(1)*g*sin(pi*x/L)+c(2)*g*sin(2*pi*x/L); %n=2
%vval=(c(1)*g*sin(pi*x/L)+c(2)*g*sin(2*pi*x/L)+
%c(3)*g*sin(3*pi*x/L)); %n=3
vval=g*(c(1)*sin(pi*x/L)+c(2)*sin(2*pi*x/L)+
%c(3)*sin(3*pi*x/L)+c(4)*sin(4*pi*x/L)+
%c(5)*sin(5*pi*x/L)); %n=5
end

%This function gives the value of the inner minimum.
%The equation below is \eta at the final time plus z(0)'*z(0).
val1 = ( y(9, 50) + (y(1:4,1))'*(y(1:4,1)))*1000;
%We multiply val1 by 1000 to speed up the code.
end

%Continuous Time Example 1
%This M-file optimizes over the perturbations dt.
%This gives the overall minimum.

%This gives the overall minimum for n=1.
%function [dt,val2] = innermin2
%This gives the overall minimum for n>1.
function val2 = innermin2(d)

dt0=[1 2.5]';
%The line below optimizes the parameter
```

```
%vector of incipient perturbations.
%innermin=@(dt)(innermin1(dt)); %For n=1
innermin=@(dt)(innermin1(d,dt)); %For n>1
options=optimset('LargeScale','off');
[dt,val2,exitflag,output] = fmincon(innermin,dt0,[],[],[],[],
[1,2]',[2,4]',[],options);
end

%Continuous Time Case
%Example 1
%Example 2
%This function optimizes d from innermin1.m.
%This function allows us to find the optimal
%parameter c of v.

function [d,val3] = outm %Outer maximization over d.

%This is an initial guess for d.
%d0=pi/2; %n=2
%d0 = [2.5 2.5]'; %n=3
d0 = [2.5 2.5 2.5 2.5]'; %n=5
inner1=@(d)(-innermin2(d));%This is the objective function.
options=optimset('LargeScale','off','Display','iter');
%For v with 2 parameters.
%[d, val3, exitflag, output] = fmincon(inner1,d0,[],[],[],[],
%0,pi,[],options) %n=2
%For v with a vector of 3 parameters.
%[d, val3, exitflag, output] = fmincon(inner1,d0,[],[],[],[],
%[-1 -1]',[5 5]',[],options) %n=3
%For v with a vector of 5 parameters.
[d, val3, exitflag, output] = fmincon(inner1,d0,[],[],[],[],
%[-1 -1 -1 -1]',[5 5 5 5]',[],options) %n=5
end

%Continuous Time
%Example 1
%Example 2
%Produces Figures 4.2,4.3,4.4,4.5,4.6
%Produces Figures 4.7,4.8,4.9,4.10,4.11
%Plot for optimal proper test signal v*
```

```
n=5; %We will also try n=5.  This is the dimension of v.
L=4; %Final time.
g=sqrt(2/L);
x=linspace(0,4,50);

%[dt,val2]=innermin2 %For n=1
[d,val3] = outm %For n>1

%The following is the computed value for vector c
%based on our optimal d.
%c2=[cos(d(1)); sin(d(1))] %For n=2.
% norm(c3)=1 based on the parameterization used.
% c3=[cos(d(1))
%    sin(d(1))*cos(d(2))
%    sin(d(1))*sin(d(2))]

% norm(c5)=1 based on the parameterization used.
c5=[cos(d(1))
   sin(d(1))*cos(d(2))
   sin(d(1))*sin(d(2))*cos(d(3))
   sin(d(1))*sin(d(2))*sin(d(3))*cos(d(4))
   sin(d(1))*sin(d(2))*sin(d(3))*sin(d(4))]

%Below is the form of the test signal, \bar{v}.
%vval1=g*sin(pi*x/L); %n=1
%vval2=c2(1)*g*sin(pi*x/L)+c2(2)*g*sin(2*pi*x/L); %n=2
%vval3=c3(1)*g*sin(pi*x/L)+c3(2)*g*sin(2*pi*x/L)+
%c3(3)*g*sin(3*pi*x/L); %n=3
vval5=g*(c5(1)*sin(pi*x/L)+c5(2)*sin(2*pi*x/L)+
c5(3)*sin(3*pi*x/L)+c5(4)*sin(4*pi*x/L)+
c5(5)*sin(5*pi*x/L)); %n=5

val3=-val3;
%The optimal proper test signal is v*=\bar{v}/sqrt(F),
%where F is the value of the overall minimum.
%vval2, vval3, vval5 are test signal form.
% v1=sqrt(1000)*vval1/sqrt(val2);
% norm(v1/vval1)
% v2=sqrt(1000)*vval2/sqrt(val3);
% norm(v2/vval2) %The norm of v*.
% v3=sqrt(1000)*vval3/sqrt(val3); %Test signal with scaling.
% norm(v3/vval3);
v5=sqrt(1000)*vval5/sqrt(val3); %Test signal with scaling.
norm(v5/vval5)

xlabel('Time')
ylabel('Test Signal, v')
title('Plot of the optimal proper incipient test signal v*')
hold all
% plot(x,v1,'k')
% plot(x,v2,'g')
% plot(x,-v3,'r')
 plot(x,-v5,'b')


%Continuous Time
%Example 1
%Optimization landscape when d is two dimensional
%Surface plot of z=innermin2(d)

x=0:0.2:3.0; y=0:0.2:3.0; %Set up x and y as vectors
[X,Y]=meshgrid(x,y); %Form the grid for plotting
z=zeros(16,16); %Initializes z
for i=1:16,
    for j=1:16,
        d=[x(i),y(j)]';
        %Function to plot; creates the height
        z(i,j)=innermin2(d);
    end
end
title('3-D Plot of innermin2(d)')
hold all
colormap(jet); %Set colors
surf(X,Y,z); %Draw surface
axis([0 3.5 0 3.5 -1 -0.1]) %Set up axes

%Continuous Time Second Example
%Solves inner minimum of the system,
%which we call 'val1', using bvp4c.
%This gives the solution to the boundary value problem.
%Untransformed Problem in paper entitled
%"Active incipient fault detection in
```

```
%continuous time systems with multiple faults"

%This is the innermost min for n=1.
function val1 = innermin11(dt)
%This is the innermost min for n>1.
%function val1 = innermin11(d,dt)

%We take norm(c)=1 and use the following parameterization.
%c=[cos(d(1)); sin(d(1))]; %For n=2
%For v 3-dimensional.
% c=[cos(d(1))
%    sin(d(1))*cos(d(2))
%    sin(d(1))*sin(d(2))];

%We take norm(c)=1 and use the following parameterization.
%For v 5-dimensional.
% c=[cos(d(1))
%    sin(d(1))*cos(d(2))
%    sin(d(1))*sin(d(2))*cos(d(3))
%    sin(d(1))*sin(d(2))*sin(d(3))*cos(d(4))
%    sin(d(1))*sin(d(2))*sin(d(3))*sin(d(4))];

L=4; %\omega=L
solinit = bvpinit(linspace(0,L,12),@bvp2init);
options = bvpset('Stats','on','RelTol',1e-7);

%The following line integrates the system
%of differential equations on the interval [0,\omega]
%subject to two point boundary conditions.
sol = bvp4c(@bvp2,@bvp2bc,solinit,options);

t=linspace(0,L,50);
y=deval(sol,t);

% figure(1)
% plot(t,y(1:4,:))
% xlabel('Time')
% ylabel('State Variable z')
% title('Solutions for z(0)')
% legend('x1','x2','xhat1','xhat2')
% figure(2)
% plot(t,y(9,:))
% xlabel('Time')
% ylabel('Value of \eta')
% title('Solutions for \eta')

function yprime = bvp2(x,y)
%The components of y correspond to the original variables
%as y(1)=z, y(2)=\lambda, y(3)=\eta.
t1=0.3; %\theta_1
t2=4.0; %\theta_2
%t3=1; %\theta_3
% dt(i)=\delta\theta_i %Perturbations

A=[-(t1) -t2 0 0; 1 -(t1) 0 0; 0 0 -(t1+dt(1)) -(t2+dt(2));
0 0 1 -(t1+dt(1))]; %\tilde{A} with 2 parameters
%A=[-(t1)*(t3) -t2 0 0; 1 -(t1) 0 0;
%0 0 -(t1+dt(1))*(t3+dt(3)) -(t2+dt(2));
%0 0 1 -(t1+dt(1))]; %\tilde{A} with 3 parameters
B=[0;1;0;1]; %\tilde{B}
C=[1 0 -1 0]; %\tilde{C}
M=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1]; %\tilde{M}

%This accounts for the first four rows of the first column.
z=y(1:4);
lam=y(5:8);
%eta=y(9,1);
vval=v(x); %For n=1
%vval=v(c,x); %For n>1

yprime = [A*z+B*vval-0.5*M*M'*lam
          -A'*lam-C'*C*z
          0.25*lam'*M*M'*lam+0.5*z'*C'*C*z];
end

function res = bvp2bc(ya,yb)
%The bvp is solved with RES=0.
%The time interval is [0 L] and the boundary conditions are:
%
%    \lambda(0)=-0.5z(0), \lambda(L)=0, \eta(0)=0.
%ya corresponds to the initial time,
%yb corresponds to the final time.
```

```
res = [ya(5:8,1)+0.5*ya(1:4,1)
       yb(5:8,1)
       ya(9,1)];
end

function yinit = bvp2init(x)
%This is the initial guess for the solutions:
%z, \lambda, r, and q:
yinit = [0.5 0 0.25 0.5 0.5 0 1 1 0]';
end

function vval = v(x) %For n=1
%function vval = v(c,x) %For n>1
L=4; %\omega=L
g=sqrt(2/L);
%This is the form of the test signal v
%and the optimization function.
vval=g*sin(pi*x/L); %n=1
%vval=c(1)*g*sin(pi*x/L)+c(2)*g*sin(2*pi*x/L); %n=2
%vval=(c(1)*g*sin(pi*x/L)+c(2)*g*sin(2*pi*x/L)+
c(3)*g*sin(3*pi*x/L)); %n=3
%vval=g*(c(1)*sin(pi*x/L)+c(2)*sin(2*pi*x/L)+
c(3)*sin(3*pi*x/L)+c(4)*sin(4*pi*x/L)+c(5)*sin(5*pi*x/L)); %n=5
end

%This function gives the value of the inner minimum.
%The equation below is \eta at the final time plus z(0)'*z(0).
val1 = ( y(9, 50) + (y(1:4,1))'*(y(1:4,1)))*1000;
%We multiply val1 by 1000 to speed up the code.
end

%Continuous Time Second Example
%This M-file optimizes over the perturbations dt.
%This gives the overall minimum.

%This gives the overall minimum for n=1.
function [dt, val2] = innermin22
%This gives the overall minimum for n>1.
%function val2 = innermin22(d)
dt0 = [0 1.5]';
%dt0=[0 1.5 0]';
```

```
%The line below optimizes the parameter
%vector of incipient perturbations.
innermin=@(dt)(innermin11(dt)); %For n=1
%innermin=@(dt)(innermin11(d,dt)); %For n>1
options=optimset('LargeScale','off');
[dt,val2,exitflag,output] = fmincon(innermin,dt0,[],[],[],[],
[-0.1,1]',[0.1,2]',[],options); %For 2 parameters
%[dt,val2,exitflag,output] =fmincon(innermin,dt0,[],[],[],[],
[-0.1,1,-0.5]',[0.1,2,0.5]',[],options);  %For 3 parameters
end
```