ABSTRACT

SWEETINGHAM, KELLY A. Auxiliary Signal Design for Fault Detection in Nonlinear Systems. (Under the direction of Professor S. L. Campbell).

Recently, research has developed in the area of active fault detection and model identification algorithms for linear systems. These algorithms compute an auxiliary input signal which guarantees fault detection, assuming a bounded noise. This dissertation addresses the question of when the previous linear theory can be applied to nonlinear systems. Several case studies are presented to verify that linearizations can in fact produce results in the nonlinear case. Two results are proven about the use of linearizations. The first result gives a parameter β to scale the entire problem. Using this parameter, the scaled auxiliary signal, along with a scaled noise bound, will guarantee fault detection in the nonlinear problem. The second result shows how to compute the acceptable noise bound for the nonlinear problem using the exact auxiliary signal from the linearized problem. Also presented is a computational study to verify these two results. Two secondary projects are also presented in this dissertation. The first is a comparison of two different linear algorithms used to compute the optimal auxiliary signal. The second is a port of one of the existing pieces of fault detection software into Matlab. This new software is included, as well as a discuss of the complications of the port.

Auxiliary Signal Design for Fault Detection in Nonlinear Systems

by
Kelly Sweetingham

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

Applied Mathematics

Raleigh, North Carolina

2008

APPROVED BY:

Dr. R. Smith	Dr. H. Tran
Dr. S. L. Campbell Chair of Advisory Committee	Dr. N. Medhin

DEDICATION

For my grandfather, Dr. Willard D. Klimstra...I know how proud he would have been of me.

BIOGRAPHY

Kelly Sweetingham was born in Tarzana, California in 1980. She is the daughter of Kay Sweetingham. She moved to Cary, North Carolina in the summer of 1990. After graduating from Cary High School in 1998, she accepted a scholarship to play soccer at the University of North Carolina at Charlotte. In the summer of 2000, Kelly transferred to The College of Charleston in South Carolina, where she continued her soccer career. Kelly started out as a Political Science major, but quickly became interested in Mathematics and Psychology. She graduated from The College of Charleston in 2002 with a Bachelors Degree in Mathematics and Psychology. In 2004, she graduated from Auburn University in Alabama with a Master's of Science in Mathematics. She began pursuing her PhD in Applied Mathematics at North Carolina State University in the Fall of 2004. She spent the summers of 2005 and 2006 interning at NAVSEA in Philadelphia. Upon graduation, she plans to move to Fairfax VA to work in the area of signal processing.

ACKNOWLEDGMENTS

First and foremost, I owe a huge amount of gratitude to my advisor, Dr. Stephen Campbell. Without his guidance and encouragement, I would not have been able to accomplish any of this.

I am also grateful to Dr. Hien Tran, Dr. Ralph Smith and Dr. Negash Medhin for serving on my committee. I sincerely appreciate the effort they put in to ensure that I produced a quality piece of work.

I must thank Dr. Kimberly Drake and NAVSEA, as well as the NSF, for providing me the support to do my research without the added pressure of teaching. Also, to Dr. Drake for giving me the opportunity to expand my skills as a summer intern.

Finally, to my friends and family...I couldn't have done it without you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background and Motivation

Fault detection and model identification algorithms are becoming increasingly important as technology expands and the ability to create accurate mathematical models improves. Fault detection and model identification is the process of diagnosing whether a system is working within normal specifications and, if not, identifying where the fault is occurring. It is important to be able to detect faults in the system because it can save cost and man power. If a fault detection algorithm identifies a fault in a particular part before the prescribed replacement time, a catastrophic failure can be avoided. This is imperative because, if a failure was to occur, it could require an entire system shut down, which costs a company time and money to repair and re-start. On the other side, if there is no fault in the system and it is the replacement time for a certain part, a company can save money by waiting to replace the part until it has actually become unusable. Clearly, as systems become increasingly complex, the ability to use human interaction to detect faults will become more difficult. Isermann, in [3], gives an extremely in depth analysis of the history and existing research in the area of fault detection.

For the remainder of this chapter, a more in depth discussion of fault detection is presented, as it relates to this dissertation. Section 1.1.1 reviews the two types of fault detection methods. Then, in section 1.2, the generic optimal control problem

and solution are presented. This control problem is used to design the minimum energy detection signal, which is presented in section 1.3.

1.1.1 Passive vs. Active Fault Detection

There are two approaches for finding faults in a system. The first is a passive approach, where there is no direct action on the system. Data from sensors and outputs are monitored and a decision is made based on this data. Typically, there is considered to be a fault if the values computed from the data fall outside some statistically acceptable range, or if an observer has been constructed, if the residual between the observer and outputs falls outside some threshold. This type of fault detection has been widely studied. We discuss some representative examples [4, 5, 6].

Bisiacco and Valcher [5] investigate the observer-based fault detection problem for 2-D state space models with and without disturbances. They consider a 2-D discrete model and construct a dead-beat observer and residual generator which only provide a good state estimator during the correct system functioning. The residual generator alone cannot function as a fault identifier due to the residual memory of the system contained in the generator. It must first go through a transient phase. In [5], they resolve the system memory issue by introducing additional constraints on the residual generator's transfer matrix and the way a signal fault influences the residual components. Specifically, they generate conditions under which the residual signal is independent from the input [5]. Bisiacco and Valcher also look at the additive noise situation for 2-D discrete models. In this case, standard dead-beat observers cannot produce an accurate state estimator, even when no fault has occurred. To overcome this obstacle, they introduce 2-D deadbeat observers which, along with satisfying normal observer conditions, must also be insensitive to disturbances. These types of observers are called deadbeat unknown input observers. Once this type of observer is constructed, the fault detector and isolator is computed as in the case without noise.

Kinnaert et al also investigate the use of residual generators, but using continuous state space models and model uncertainty [6]. Noting that, using the typical approaches to compute a residual generator, the stability of the plant is not a necessary

condition due to the assumption that estimated plant model exactly coincides with true plant model, [6] looks at what happens when this assumption is not true. They show that it is impossible to build a residual generator for an unstable plant because the unstable modes cannot be made unobservable from the residual when there is uncertainty in the model. However, by using a stabilizer control, this issue is resolved and a residual generator can be built.

While the above papers use residuals to determine faults, [7] discusses two generations of operational fault detection (OFD) algorithms which are based on statistical thresholds for wireless networks. This type of network is interesting because it has numerous base stations distributed over a wide geographical area and the fault scenarios require complex detection and recovery schemes [7]. The first generation OFD are out-dated due to burdensome human interaction. Test quantities (TQ's) are predetermined call-processing observations. First generation OFD's require a technician to identify a threshold violation, which is an event when a TQ exceeds or goes below a preset threshold value or duration. There can be violations in values, duration, or probability distribution (the latter does not require human intervention, it uses a priori knowledge of the probability distribution). The threshold values are typically extreme values of the TQ during healthy operation of the system. The procedure includes running the OFD on the healthy system for approximately 2 weeks to identify worst case scenarios and help determine the threshold values. Then, a fault is considered to happen if a base station has no call processing activity for a time duration greater than the specific threshold. However, if the current TQ is worse than the worst case TQ and the threshold crossing is not associated with a fault, then the threshold is set to present TQ. For the second generation OFD, operator intervention for setting the threshold values is eliminated [7]. In this case, the OFD has the ability to adaptively understand and react to help mimic the technician. There are two different algorithms to choose from for characterizing good health [7]. The first one is the Learning Algorithm, which consists of two stages, learning and detecting. in the learning stage, the algorithm is presented with a time sequence of TQ's to build a baseline profile by computing expected values. The baseline characterizes the offered load at different times. In the detection phase, the algorithm is "suspicious" of a

fault when an observed value falls below a certain function β of the baseline value. When the probability of truant traffic goes up, the probability of an actual fault goes up, where truant traffic is when the system is working properly but traffic is light and gives the semblance of a problem. When the probability of a fault exceeds a certain confidence level, the algorithm declares a fault. The second algorithm is a correlation based algorithm [7]. There is no learning phase in this process. Instead, the estimated offered load is obtained by looking at the neighboring sectors. It is assumed that there is some appreciable correlation between neighbors. Note that this algorithm works even for moderate correlations. The algorithm tracks the long-term mean load and running covariances for each target cell. In order for a fault to be declared, a significant difference between the mean variance and computed expected variance must last for a specified duration.

In [4], faults are diagnosed via a causal fault-symptom tree. Isermann sets up a knowledge based fault detection problem with two parts. The first part consists of analytic knowledge and the second is heuristic knowledge. The analytic knowledge comes from [4]

- theoretical modeling from physical laws,
- estimation for parameters and state variables,
- normal process behavior and,
- fault statistics, if quantifiable.

Whereas, the heuristic knowledge comes from

- fault trees,
- fault statistics, if qualitively known.

A fault tree is a connection of symptoms and faults. Fault detection is first based on the observed symptoms and available fault trees. If there is determined to be a fault, then a fault decision is done using process history and fault statistics. Then the type of fault, size and cause are suggested.

All of the above deal with either discrete or continuous ODE's. In [8] a passive approach to fault detection is discussed for a nonlinear coupling of differential algebraic equations (DAE's) and ordinary differential equations (ODE's), known as nonlinear differential algebraic systems. Since DAE's are naturally singular, the standard techniques for solving ODE's typically don't carry over well to differential algebraic systems. In order to perform fault analysis on the system, there must be a decoupling of the DAE's and ODE's, as well as assumptions on the uncertainties (unstructured and bounded). Polycapou et al model the faults as nonlinear functions of the state and algebraic variables vectors and input vectors.

Although there is clearly a lot of research in this type of fault detection, one of the major flaws of this approach is that normal operation of the system may mask faults. This can occur when the portion of the system containing the fault is inactive (i.e. a fault in the brakes would not be diagnosed simply by monitoring a running car), or control action masks the fault, or when the damage is to the sensor itself. Damaged sensors may make the previously constructed fault detection procedures ineffective for the new situation.

Active fault detection, on the other hand, inputs a signal into the system in order to determine if there has been a fault. It appears that the first area to develop active fault detection methods, or process model-based fault detection methods, was in aerospace systems and chemical plants [3]. In this type of method, the output of the system, after applying the signal, is measured and the algorithm makes a decision whether there is a fault. An advantage of this approach is there is no need to constantly monitor the system which the passive approach requires. Also, this approach can determine faults that are masked by the control system. The drawback to the model identification approach is that the models for the normal and faulty system must be sufficiently accurate or the algorithm might produce a false alarm [2]. This approach has not been as widely studied as the passive one. In the following, a minimal overview of previous active fault detection is discussed. Then, in Section 1.3 two specific active fault detection algorithms which are directly related to the research of this dissertation are described in detail.

In [9], Niemann attempts to extend fault detection results from closed loop systems

to open loop systems as well as closed loop systems with feedback controllers. The main difference in fault detection in open loop systems is the stability aspect. Starting with a closed loop system, he derives an open loop system by removing the feedback controller. Then the control input needs to be applied since the input injection point no longer exists without the feedback. A transfer function matrix is derived between the input vector and the residual vector. This transfer matrix, called the open loop fault signature matrix is equivalent to the fault signature matrix in the closed loop system [9]. Setting up the open loop system in this manner, [9] shows that it is possible to use the same active fault detection method as on the closed loop system, with the only restriction being that the maximal number of faults occurring simultaneously is bounded by the number of control signals minus one.

1.2 Mathematical Background

In this section, we give a review of the optimal control problem which will be used in this thesis. In order to give the full picture of the minimum energy detection signal (MEDS) algorithm, developed by Campbell and Nikoukhah [10], we must first review the optimal control problem in general. This is due to the fact that the actual design of the signal is based on optimal control theory.

1.2.1 The Optimal Control Problem

As it will be seen in Section 1.3, the detection signal is actually the solution to an optimal control problem. The goal of the algorithm is to build an input signal, referred to as the control, which will affect the system in some desired way. Thus, it is of importance that the general optimal control problem is understood.

We begin with a first order system of ordinary differential equations given by

$$\dot{x}(t) = f(x, u, t), \tag{1.1}$$

where $x(t) \in R^n$ is the state of the system and $u \in R^m$ is our input, or control. There

is typically a cost, or performance measure, of the system given by

$$J = h(x(t_0), t_0) + h_f(x(t_f), t_f) + \int_{t_0}^{t_f} g(x(t), u(t), t)\, dt. \tag{1.2}$$

In (1.2), $[t_0, t_f]$ is the time interval of interest, h, h_f and g represent the system operation at the initial time, final time and intermediate times, respectively. If we call $u^*(t)$ the admissible control and $x^*(t)$ the admissible trajectory, then the goal of the optimal control problem is to find an admissible control, $u^*(t)$, that causes the system (1.1) to follow an admissible trajectory, $x^*(t)$, that minimizes the cost functional (1.2).

In a sub-problem of the design of the detection signal, the initial and final times (t_0 and t_f) are fixed and evaluation at the final time is not considered, i.e. $h_f \equiv 0$. Thus, the minimization problem is

$$\min_u \qquad J = h(x(t_0), t_0) + \int_{t_0}^{t_f} g(x(t), u(t), t)\, dt \tag{1.3a}$$

$$\text{subject to} \qquad \dot{x} = f(x, u, t). \tag{1.3b}$$

To solve this problem, we will use the Calculus of Variations. We would like to rewrite (1.3a) to include the constraint (1.3b) so we introduce a Lagrangian multiplier γ and write (1.3) as

$$\widehat{J} = h(x(t_0), t_0) + \int_{t_0}^{t_f} (g + \gamma^T (f - \dot{x}))\, dt, \tag{1.4}$$

where $\gamma = \gamma(t)$ since (1.3b) holds for all t. Now, we will introduce a function $H = g(x, u, t) + \gamma^T f(x, u, t)$, called a Hamiltonian. Rewriting (1.4) using the Hamiltonian, we have

$$\widehat{J} = h(x(t_0), t_0) + \int_{t_0}^{t_f} (H(x, u, t) - \gamma^T \dot{x})\, dt. \tag{1.5}$$

Optimal control law states that the differential of $\widehat{J}, d\widehat{J}$ must be equal to zero.

From Leibniz's Rule[1], we know that

$$\begin{aligned} d\widehat{J} &= h_x dx|_{t_0} + (H - \gamma^T \dot{x})dt|_{t_f} - (H - \gamma^T \dot{x})\ dt|_{t_0} \\ &\quad + \int_{t_0}^{t_f} H_x \delta x + H_u \delta u - \gamma^T \delta \dot{x} + (H_\alpha - \dot{x})^T \delta\gamma\ dt. \end{aligned} \tag{1.6}$$

We must resolve the issue of the dependence of the differentials dx and dt. To do this, we note that, if we fix a point $t = \bar{t}$, then $dx(\bar{t}) = \delta x(\bar{t}) + \dot{x}(\bar{t})d\bar{t}$. Recall that in our sub-problem, we have a fixed initial and final time. Thus, there is no variation in t_0 and t_f, i.e. $dt_0 = 0$ and $dt_f = 0$. Hence

$$dx(t_0) = \delta x(t_0) \tag{1.7a}$$

$$dx(t_f) = \delta x(t_f). \tag{1.7b}$$

Using (1.7) in (1.6), $d\widehat{J}$ becomes

$$d\widehat{J} = h_x \delta x|_{t_0} + \int_{t_0}^{t_f} H_x \delta x + H_u \delta u - \gamma^T \delta \dot{x} + (H_\gamma - \dot{x})^T \delta\gamma\ dt \tag{1.8}$$

Now, we note in (1.8) that we have a product including a derivative inside the integral ($\gamma^T \delta \dot{x}$). Thus, we can use integration by parts to eliminate it:

$$-\int_{t_0}^{t_f} \gamma^T \delta \dot{x} dt = \gamma^T \delta x|_{t_0} - \gamma^T \delta x|_{t_f} + \int_{t_0}^{t_f} \dot{\gamma}^\delta x\ dt. \tag{1.9}$$

Substituting (1.9) in (1.8), the final $d\widehat{J}$ becomes

$$d\widehat{J} = (h_x + \gamma^T)\delta x + \int_{t_0}^{t_f} (H_x + \dot{\gamma}^T)\delta x + H_u \delta u + (H_\gamma - \dot{x})^T \delta\gamma\ dt. \tag{1.10}$$

As stated earlier, $d\widehat{J}$ must equal zero. However, the variations of $\delta x, \delta u$ and $\delta\gamma$ are functions of t. Therefore, the coefficients of these terms must equal zero. Thus,

[1]Leibniz's Rule: If $x(t) \in R^n$ is a function of t and $J(t) = \int_{t_0}^{t_f} \alpha(x(t), t)dt$, where $J(\cdot)$ and $\alpha(\cdot)$ are real scalar functions, then $dJ = \alpha(x(t_f), t_f)dt_f - \alpha(x(t_0), t_0)dt_0 + \int_{t_0}^{t_f} [\alpha^T(x(t), t)\delta x]dt$.

the optimal solution to minimize J must adhere to the following requirements.

$$H_\gamma = \dot{x} \tag{1.11a}$$
$$H_x = -\dot{\gamma}^T \tag{1.11b}$$
$$H_u = 0 \tag{1.11c}$$
$$h_x(t_0) = -\gamma^T(t_0). \tag{1.11d}$$

We call (1.11a) the state equation, (1.11b) the costate equation, (1.11c) the algebraic equation, and (1.11d) is a boundary condition.

1.3 Fault Detection Algorithms

In Section 1, we give an overview of the various different techniques for fault detection. In the following section we give detailed descriptions of two specific active fault detection algorithms developed by Campbell and Nikoukhah [10]. Section 1.3.1 discusses the Minimum Energy Detection Signal (MEDS) Algorithm, while Section 1.3.2 presents the Direct Optimization (DO) approach. Both sections rely heavily on [10] and Section 1.3.1 also relies on [11] and a more in depth discussion can be found in there. We will not present the model identification process since this dissertation does not discuss this issue. Again, an in depth discussion can be found in [10].

1.3.1 Minimum Energy Detection Signal Algorithm

In the previous section, we introduced the generic optimal control problem to give us an introduction the the Minimum Energy Detection Signal (MEDS) Algorithm. This control problem is central to the development of the MEDS Algorithm that follows. The linear case has been developed for both additive and model uncertainty. However, the nonlinear results presented in Chapter 4 consider only additive noise.

Although this is a multi-model algorithm, for simplicity we consider only two models, $i = 0$ being the properly working model and $i = 1$ being the faulty model. We assume that we have linear, constant coefficients with additive uncertainty. The

models are of the form

$$\dot{x}_i = A_i x_i + B_i v + M_i \mu_i \tag{1.12a}$$

$$E_i y_i = C_i x_i + N_i \mu_i. \tag{1.12b}$$

v is our input signal and μ_i is the noise associated with the properly working system ($i = 0$) and the faulty system ($i = 1$). We also assume that N_i, $i = 0, 1$, has full row rank and E_i, $i = 0, 1$ has full column rank. For additive uncertainty we have $E_i = I$. We are assuming that there is no extra control input into the system, i.e. all inputs are used as an auxiliary signal for fault detection. We also note that we are on a time interval of $[0, T]$. For model uncertainty, we look at some time interval $[0, \omega]$, where $\omega \leq T$. However, for additive noise, we use the entire interval.

Let $\mathcal{A}_i(v)$ be the output set of model i. Perfect fault detection would imply that $\mathcal{A}_0(v) \cap \mathcal{A}_1(v) = \emptyset$. Without a bound on the uncertainty, or noise, any output would be possible from either model and perfect model identification would be impossible [2]. Thus, we assume that the noise in model i is bounded in the L^2 norm by the noise measure

$$\mathcal{S}_i(x_i(0), \mu_i) = x_i(0)^T P_{i0}^{-1} x_i(0) + \int_0^T |\mu_i(t)|^2 \ dt < \gamma. \tag{1.13}$$

For purposes of discussion, we take $\gamma = 1$. Once the noise is bounded, we can consider the outputs of the models. Define $\mathcal{L}_i(f) = \int_0^T e^{A_i(t-s)} f(s) ds$ so that $\mathcal{L}_i(f)$ is the solution of $\dot{z} = A_i z + f$, $z(0) = 0$. Let ξ_i be the free initial condition for ODE (1.12a). Then

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

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

Thus, the output sets, $\mathcal{A}_i(v)$, are translates of open sets by $\bar{y}_i$ and they are affinely dependent upon the detection signal, v [2]. The goal of the algorithm is to find the minimum proper signal, where a proper signal is one which separates the two output sets. This signal is then input into the system during a short test period and the

output is measured. Based on the measurement, a decision is made whether the system is normal or faulty.

If (x_0, μ_0, x_1, μ_1) satisfies the models (1.12) and the noise bound (1.13), then a proper detection signal, v, ensures that the output sets of the models will be distinct. Alternatively if (x_0, μ_0, x_1, μ_1) satisfy the models and v is a proper detection signal, but the outputs are still not distinct, then (x_0, μ_0, x_1, μ_1) must not satisfy the noise bound (1.13) for $i = 0, 1$ [2]. Thus, if (x_0, μ_0, x_1, μ_1) satisfies the models and v is a proper detection signal, but an output, y, is still in both output sets, it must be that

$$\max\{\mathcal{S}_0(x_0(0), \mu_0), \mathcal{S}_1(x_1(0), \mu_1)\} \geq 1. \tag{1.14}$$

As this must be true for all possible (x_0, μ_0, x_1, μ_1) that satisfy the models but do not have distinct outputs, it is sufficient to ensure it is true for the minimum of them. Thus, we have the following characterization of a proper v [2]

$$\sigma(v) = \min_{x_i, \mu_i, y_0 = y_1} \max\{\mathcal{S}_0(x_0(0), \mu_0), \mathcal{S}_1(x_1(0), \mu_1)\} \geq 1. \tag{1.15}$$

Thus, the goal of the algorithm is to find the minimum proper signal v such that (1.15) holds. Figure 1.3.1 gives a pictorial of the effect of the optimal signal v.

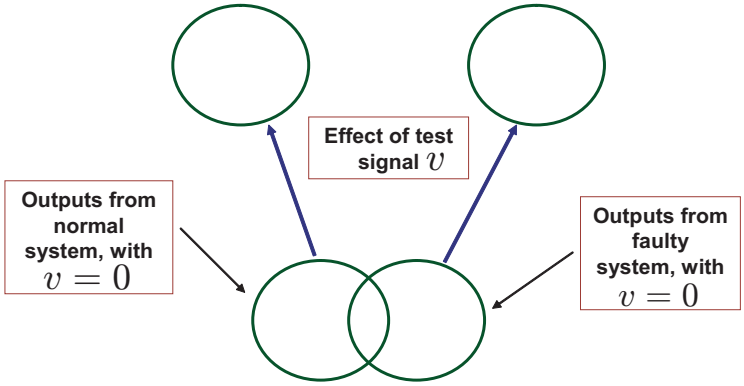


Figure 1.1: Disjoint output sets.

In order to compute v, we note that we can write (1.15) as

$$\sigma(v) = \max_{\beta \in [0,1]} \phi_\beta(v), \tag{1.16}$$

where

$$\phi_\beta(v) = \min_{x_i,\mu_i,y_0=y_1} \beta\mathcal{S}_0(x_0,\mu_0) + (1-\beta)\mathcal{S}_1(x_1,\mu_1). \tag{1.17}$$

We would like to combine both the properly working model and faulty model into one large ODE. To do this, we use the notation

$$x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}, \mu = \begin{bmatrix} \mu_0 \\ \mu_1 \end{bmatrix}, A = \begin{bmatrix} A_0 & 0 \\ 0 & A_1 \end{bmatrix}, B = \begin{bmatrix} B_0 \\ B_1 \end{bmatrix}, M = \begin{bmatrix} M_0 & 0 \\ 0 & M_1 \end{bmatrix} \tag{1.18a}$$

$$D = F_0D_0 + F_1D_1, C = \begin{bmatrix} F_0C_0 & F_1C_1 \end{bmatrix}, N = \begin{bmatrix} F_0N_0 & F_1N_1 \end{bmatrix} \tag{1.18b}$$

$$P_\beta^{-1} = \begin{bmatrix} \beta P_{0,0}^{-1} & 0 \\ 0 & (1-\beta)P_{1,0}^{-1} \end{bmatrix}, J_\beta = \begin{bmatrix} \beta & 0 \\ 0 & 1-\beta \end{bmatrix}, \tag{1.18c}$$

where

$$F = \begin{bmatrix} F_0 & F_1 \end{bmatrix} = \begin{bmatrix} E_0 \\ E_1 \end{bmatrix}^\perp. \tag{1.19}$$

Note that $Z^\perp$ is the maximal rank right annihilator of the matrix Z and $Z_\perp$ is the corresponding left annihilator. Now (1.16) and (1.17) can be reformulated into one larger problem.

$$\phi_\beta = \inf_{\mu,x} \left\{ x(0)^T P_\beta^{-1} x(0) + \int_0^T \mu^T J_\beta \mu \, dt \right\} \tag{1.20}$$

subject to

$$\dot{x} = Ax + Bv + M\mu \tag{1.21a}$$

$$0 = Cx + Dv + N\mu. \tag{1.21b}$$

We consider a cost function for v of the form

$$\delta(v) = \xi(T)^T W \xi(T) + \int_0^T |v|^2 + \xi^T U \xi \, dt \tag{1.22a}$$

$$\dot{\xi} = F\xi + Gv, \ \xi(0) = 0. \tag{1.22b}$$

U and W are semi-positive definite matrices (with no dependence on s) and F and G are design parameters. Most often, to have (1.22b) represent normal operating behavior without uncertainties, we take $F = A_0$ and $G = B_0$. Then $\xi(t)$ is an a-priori estimate of $x_0(t)$ and penalizing ξ amounts to penalizing the effect of v to the system during the test period without a fault.

Using the left annihilator of N, we assume that $N_\perp^T J_\beta N_\perp > 0$ for some $\beta \in [0,1]$ and we let $\mathcal{B}$ be the set of such β. Restating the problem in terms of $\mathcal{B}$, it becomes

$$\min_v \delta(v) \tag{1.23a}$$

subject to

$$\max_{\beta\in\mathcal{B}} \phi_\beta(v) \geq 1. \tag{1.23b}$$

Now we define

$$\lambda_\beta = \max_{v\neq 0} \frac{\phi_\beta(v)}{\xi(T)^T W \xi(T) + \int_0^T |v|^2 + \xi^T U \xi \; dt} \tag{1.24}$$

$$= \max_{v\neq 0} \frac{\phi_\beta(v)}{\delta(v)}. \tag{1.25}$$

Rearranging terms, we get

$$\lambda_\beta \delta(v) = \max_{v\neq 0}. \tag{1.26}$$

If we substitute (1.20) and (1.22a) into (1.26) and rearranging terms, what we need to solve becomes

$$\max_v \inf_{\mu,x} \left\{ x(0)^T P_\beta^{-1} x(0) - \lambda \xi(T)^T W \xi(T) + \int_0^s \mu^T J_\beta \mu - \lambda(\xi^T U \xi + |v|^2) \; dt. \right\} \tag{1.27}$$

subject to (1.22b) and (1.21). Letting $\mathcal{Q}_\lambda = \text{diag}(0, \lambda U)$ and $\nu^T = \begin{bmatrix} x^T & \xi^T \end{bmatrix}^T$, (1.27) can be rewritten as

$$\begin{aligned} &\nu(0)^T \text{diag}(P_\beta^{-1}, 0)\nu(0) + \nu(s)^T \text{diag}(0, -\lambda W)\nu(s) \\ &+ \int_0^s \mu^T J_\beta \mu - \lambda |v|^2 - \nu^T \mathcal{Q}_\lambda \nu \; dt. \end{aligned} \tag{1.28}$$

Restating our goal, we would like to minimize (1.28). From Section 1.2.1, we know that a necessary condition for this problem can be expressed as the two point boundary value problem (TPBVP) (1.29). First, we will introduce some new notation. If we let

$$\bar{A} = \text{diag}(A, F), \bar{B} = \begin{bmatrix} M & B \\ 0 & G \end{bmatrix}, \bar{C} = \begin{bmatrix} C & 0 \end{bmatrix},$$

then

$$\bar{D} = \begin{bmatrix} N & D \end{bmatrix}, \Gamma_{\lambda,\beta} = \text{diag}(J_\beta, -\lambda I)$$

and we have that

$$\begin{bmatrix} \bar{Q}_{\lambda,\beta} & \bar{S}_{\lambda,\beta} \\ \bar{S}_{\lambda,\beta}^T & \bar{R}_{\lambda,\beta} \end{bmatrix} = \begin{bmatrix} \bar{B} \\ \bar{D} \end{bmatrix} \Gamma_{\lambda,\beta}^{-1} \begin{bmatrix} \bar{B} \\ \bar{D} \end{bmatrix}^T.$$

Finally, we define $\bar{\Omega}_{11} = -\bar{\Omega}_{22}^T = \bar{A} - \bar{S}_{\lambda,\beta}\bar{R}_{\lambda,\beta}^{-1}\bar{C}$, $\bar{\Omega}_{12} = \bar{Q}_{\lambda,\beta} - \bar{S}_{\lambda,\beta}\bar{R}_{\lambda,\beta}^{-1}\bar{S}_{\lambda,\beta}^T$ and $\bar{\Omega}_{21} = \bar{C}^T\bar{R}_{\lambda,\beta}^{-1}\bar{C} + \mathcal{Q}_{\lambda,\beta}$. Then the TPBVP is

$$\frac{d}{dt}\begin{bmatrix} \nu \\ \zeta \end{bmatrix} = \begin{bmatrix} \bar{\Omega}_{11} & \bar{\Omega}_{12} \\ \bar{\Omega}_{21} & \bar{\Omega}_{22} \end{bmatrix} \begin{bmatrix} \nu \\ \zeta \end{bmatrix} = H \begin{bmatrix} \nu \\ \zeta \end{bmatrix} \tag{1.29a}$$

$$V_0 \begin{bmatrix} \nu(0) \\ \zeta(0) \end{bmatrix} + V_s \begin{bmatrix} \nu(s) \\ \zeta(s) \end{bmatrix} = 0, \tag{1.29b}$$

where $V_0 = \begin{bmatrix} I & \mathrm{diag}(-P_\beta, 0) \\ 0 & 0 \end{bmatrix}$, $V_s = \begin{bmatrix} 0 & 0 \\ \mathrm{diag}(0, \lambda W) & I \end{bmatrix}$.

The optimal v and μ satisfy

$$\begin{bmatrix} \mu \\ v \end{bmatrix} = \alpha\Gamma_{\lambda,\beta}^{-1} \begin{bmatrix} \bar{D}^T\bar{R}_{\lambda,\beta}^{-1}\bar{C} & \bar{D}^T\bar{R}_{\lambda,\beta}^{-1}\bar{S}_{\lambda,\beta}^T - \bar{B}^T \end{bmatrix} \begin{bmatrix} \nu \\ \zeta \end{bmatrix} \tag{1.30}$$

where α is a to be determined scalar. We need to find λ_β which is the largest λ for which (1.29) is not well posed for some $s \in [0, T]$. However, we note that, since H is a Hamiltonian, the wellposedness can be determined by block diagonalizing as

$$SHS^{-1} = \mathrm{diag}(A_f, A_b), \tag{1.31}$$

where A_f and A_b do not have any eigenvalues with strictly positive real parts. Under our assumption, $\lambda > \lambda_\beta$ if and only if $\Psi(s)$ is invertible for all $s \in [0, T]$, where

$$\Psi(s) = V_0 S^{-1}\mathrm{diag}(I, e^{A_b s}) + V_s S^{-1}\mathrm{diag}(e^{A_f s}, I). \tag{1.32}$$

This means we need to solve

$$\dot{\Psi}_f = A_f\Psi_f, \quad \Psi_f(0) = I \tag{1.33a}$$

$$\dot{\Psi}_b = A_b\Psi_b, \quad \Psi_b(0) = I \tag{1.33b}$$

and determine if the surface

$$0 = \det\Psi(s) \tag{1.34}$$

was crossed. λ_β is the smallest λ for which the above system can be solved without an surface crossing. We then define the optimal λ^* and β^* to be

$$\lambda^* = \max_{\beta \in \mathcal{B}} \lambda_\beta. \tag{1.35}$$

In order to compute the optimal v from (1.30), we note that when we have the optimal λ and β, (1.29) has a non-zero solution. The optimal auxiliary signal is computed from (1.30) by choosing α such that $\|v\|^{-1} = \sqrt{\lambda^*}$.

1.3.2 Direct Optimization Algorithm

In the previous section, a detailed description of the MEDS algorithm is given. However, it is not the only way to compute the optimal signal. Another approach, called the Direct Optimization Algorithm, is described below. This approach allows for more general problems such as pointwise bounds, problems with additional known inputs and problems with additional design parameters on the auxiliary signal [10]. More information about this section can be found in [12]. For this section, the computations have been automated in Matlab, Maple and Sparse Optimal Control Software (SOCS) [13].

We start with the same problem set up as with the MEDS algorithm.

$$\dot{x}_i = A_i x_i + B_i v + M_i \mu_i \tag{1.36a}$$

$$y = C_i x_i + D_i v + M_i \mu_i \tag{1.36b}$$

and we need to minimize

$$\beta S_0 + (1 - \beta) S_1 \tag{1.37}$$

subject to (1.36). For the DO routine, we do this minimization directly as an optimization problem.

We start with performing a QR decomposition on $N_i^T = Q_i R_i$, where Q_i is unitary. We let

$$R_i^T = \begin{bmatrix} \bar{N}_i & 0 \end{bmatrix} \tag{1.38}$$

$$Q_i^T \mu_i = \begin{bmatrix} \bar{\mu}_i \\ \tilde{\mu}_i \end{bmatrix} \tag{1.39}$$

where $\bar{N}_i$ is invertible. Therefore,

$$N_i\mu_i = \begin{bmatrix}\bar{N}_i & 0\end{bmatrix}\begin{bmatrix}\bar{\mu}_i \\ \tilde{\mu}_i\end{bmatrix}. \tag{1.40}$$

Doing a similar decomposition of M, we let

$$M_iQ_i^{-T} = \begin{bmatrix}\bar{M}_i & \tilde{M}_i\end{bmatrix} \tag{1.41}$$

with the same partitioning as (1.38). Then

$$M_i\mu_i = M_iQ_i^{-T}Q_i^T\mu_i = \begin{bmatrix}\bar{M}_i & \tilde{M}_i\end{bmatrix}\begin{bmatrix}\bar{\mu}_i \\ \tilde{\mu}_i\end{bmatrix}. \tag{1.42}$$

Now, the system model can be reconfigured as

$$\dot{x}_i = A_ix_i + B_iv + \bar{M}_i\bar{\mu}_i + \tilde{M}_i\tilde{\mu}_i \tag{1.43a}$$

$$y = C_ix + D_iv + \bar{N}_i\bar{\mu}_i. \tag{1.43b}$$

To combine both equations for y, we note that $0 = C_1x_1 + D_1v + \bar{N}_1\bar{\mu}_1 - C_0x_0 - D_0v - \bar{N}_0\bar{\mu}_0$ and we solve this equation for $\bar{\mu}_0$ and substitute into (1.43a) to get

$$\dot{x}_0 = (A_0 - \bar{M}_0\bar{N}_0^{-1}C_0)x_0 + \bar{M}_0\bar{N}_0^{-1}C_1x_1 + B_0v + \tilde{M}_0\tilde{\mu}_0 + \bar{M}_0\bar{N}_0^{-1}\bar{N}_1\bar{\mu}_1. \tag{1.44}$$

If we let

$$x = \begin{bmatrix}x_0 \\ x_1\end{bmatrix}, A = \begin{bmatrix}A_0 - \bar{M}_0\bar{N}_0^{-1}C_0 & \bar{M}_0\bar{N}_0^{-1}C_1 \\ 0 & A_1\end{bmatrix}, B = \begin{bmatrix}B_0 \\ B_1\end{bmatrix},$$

$$M = \begin{bmatrix}\tilde{M}_0 & \bar{M}_0\bar{N}_0^{-1}\bar{N}_1 & 0 \\ 0 & \bar{M}_1 & \tilde{M}_1\end{bmatrix}, \mu = \begin{bmatrix}\bar{\mu}_0 \\ \tilde{\mu}_1 \\ \tilde{\mu}_0\end{bmatrix},$$

then the combined system is $\dot{x} = Ax + Bv + M\mu$.

Next, we have $C = \begin{bmatrix}C_0 & -C_1\end{bmatrix}$ and $N = \begin{bmatrix}0 & \bar{N}_1 & 0\end{bmatrix}$ and we let

$$H = -4\beta C^TN,$$

$$Q = 2\beta C^TC$$

and

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

There are two different setups for the problem to be input into SOCS. These formulations are for the case when the L2 norm of v is to be minimized. If the cost (1.22) is used, the equations need to be supplemented.

- Setup One:

$$(\gamma^*)^2 = \max_{v,\beta} Z(\omega) \tag{1.45}$$

subject to

$$\begin{aligned} \dot{x} &= Ax + Bv + M\mu && \text{(1.46a)} \\ \dot{\lambda} &= -Qx - \frac{1}{2}H\mu - A^T\lambda, \lambda(0) = \lambda(\omega) = 0 && \text{(1.46b)} \\ \dot{\theta} &= v^T v, \theta(0) = 0, \theta(\omega) = 1 && \text{(1.46c)} \\ \dot{Z} &= \frac{1}{2}[x^T Q x + x^T H\mu + \mu^T R\mu], Z(0) = 0 && \text{(1.46d)} \\ 0 &= R\mu + \frac{1}{2}H^T x + M^T\lambda && \text{(1.46e)} \\ & 0.01 < \beta < 0.99 && \text{(1.46f)} \end{aligned}$$

- Setup Two:

$$(\gamma^*)^{-2} = \min_{v,\beta} \int_0^\omega \|v\|^2 dt \tag{1.47}$$

subject to

$$\begin{aligned} \dot{x} &= Ax + Bv + M\mu && \text{(1.48a)} \\ \dot{\lambda} &= -Qx - \frac{1}{2}H\mu - A^T\lambda, \lambda(0) = \lambda(\omega) = 0 && \text{(1.48b)} \\ \dot{Z} &= \frac{1}{2}[x^T Q x + x^T H\mu + \mu^T R\mu], Z(0) = 0, Z(\omega) \geq 1 && \text{(1.48c)} \\ 0 &= R\mu + \frac{1}{2}H^T x + M^T\lambda && \text{(1.48d)} \\ & 0.01 < \beta < 0.99 && \text{(1.48e)} \end{aligned}$$

1.4 Chapter Summary

This dissertation represents the culmination of my work. In chapter 2 we present a study to compare the MEDS algorithm to the Direct Optimization (DO) routine, both described in Section 1.3. As a side note, the MEDS algorithm is sometime referred to as the Constrained Control (CC) algorithm and the two names will be used interchangeably. This leads us to Chapter 3, where we give several case studies which suggest that linearizations might be a feasible way to use the linear theory on nonlinear problems. In Chapter 4, nonlinear analysis is presented which gives situations where the optimal signal from the linearized system can be used on the nonlinear system. The linear MEDS algorithm was originally written in Scilab, however, there was a request to port this into Matlab. This process is given in Chapter 5. Also presented are four case studies to show that the Matlab MEDS algorithm gives the same results as the original Scilab algorithm. Chapter 6 discusses future research proposals to extend these ideas to other types of nonlinearities. Finally, in Chapter 7, concluding remarks are given, as well as a summary of the contributions of this dissertation.

Chapter 2

Comparison of Methods for Solving MEDS Algorithm

In this chapter we discuss the application of the two active fault detection algorithms described in the Introduction based on model identification to power systems. My research is concerned with design of the auxiliary signal. Thus, we focus on finding this signal. In addition to the comparison of the two approaches, we also offer comment on the performance of the algorithms on a stiff problem.

2.1 Examples

2.1.1 DC Motor

Our first example is a simple DC motor, which is used to provide rotational energy. By varying the voltage, we can control the rate at which the motor rotates and the current in the system, so our input is the voltage and our output is the rate of rotation. We assume that the magnetic field is constant. The original parameters and derivation of the model are acquired from the MATLAB Control System Toolbox [14].

Let $\omega(t)$ be the angular rate of the load and $v_{app}(t)$ the applied voltage. Let R be the resistance of the circuit and L the self-inductance of the armature created by

the current. The torque of the shaft of the motor is proportional to the current, $i(t)$, so we require a proportionality constant related to the properties of the motor [14]. Also, the back electromotive force is proportional to the angular rate of the load [14], so we obtain a second proportionality constant depending on the properties of the motor. In SI units, these two proportionality constants are equivalent [15], we denote them as K_m.

The equations we use to relate the control (voltage) to the state variables (angular rate and current) are Kirchhoff's law and Newton's law. We can write the differential equations, including the additive uncertainty and output equation as

$$\frac{di}{dt} = -\frac{R}{L}i(t) - \frac{K_m}{L}\omega(t) + \frac{1}{L}v_{app}(t) + \mu_1, \tag{2.1a}$$

$$\frac{d\omega}{dt} = \frac{K_m}{J}i(t) - \frac{K_f}{J}\omega(t) + \mu_2. \tag{2.1b}$$

$$y = \omega(t) + \mu_3 \tag{2.1c}$$

In state space we have

$$\frac{d}{dt}\begin{bmatrix} i \\ \omega \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{K_m}{L} \\ \frac{K_m}{J} & -\frac{K_f}{J} \end{bmatrix}\begin{bmatrix} i \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} v_{app}(t) + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}, \tag{2.2a}$$

$$y = \omega(t) + \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}, \tag{2.2b}$$

where we have included noise in the system as μ, while y represents the output of the model.

For the properly working model we use the parameters defined in MATLAB [14]. These can be found in Table 2.1.

When looking at the faulty model, we determine that two possible faults are the resistance in the motor (R) and the coefficient of friction (K_f). The resistance of the motor will increase if the wiring has been damaged, since the wire cannot conduct

Table 2.1: Parameter values for both the faulty and properly working models.

Parameter	Unfaulty (Model 0)	Faulty (Model 1)
R	2.0 Ohms	3.0 Ohms
L	0.5 Henrys	0.5 Henrys
K_m	0.015 $\frac{\text{Nm}}{\text{A}}$	0.015 $\frac{\text{Nm}}{\text{A}}$
K_f	0.2 Nms	0.3 Nms
J	0.02 $\frac{kgm^2}{s^2}$	0.02 $\frac{kgm^2}{s^2}$

electricity as efficiently. Thus, for our faulty model we will use a resistance of 3.0 Ohms. We can also have an increase in the coefficient of friction, representing a loss of lubricant between the shaft of the motor and any surrounding stationary parts. In this situation we assume a tolerance of 0.1 Nms, so the coefficient of friction for the faulty model is 0.3 Nms. For simplicity we assume the faults occur together. The noise coefficients are the same for both models.

In our simulation we use a test interval of 5.0 seconds. We can see from figure 2.1.1 that the two algorithms produce almost identical input signals.

We implemented the DO algorithm using SOCS. In order to implement the MEDS algorithm, we used a code provided in [10] written in Scilab. The runtime of the codes depends heavily on the code parameters (β grid size, for instance), but the SOCS runtime is on the order of a few seconds and the SciLab runtime is a couple minutes. However, as can be seen in figure 2.1.1, the two algorithms produce nearly an identical signal over the 5 second test period.

Overall, both models take a reasonable amount of time to run for this simple model. If we approximately know β^*, then we can speed up the runtime of the SciLab code by decreasing the β search interval and decreasing the number of grid points. Currently we use a β-interval of $[\epsilon, 1-\epsilon]$, for some small number ϵ, and we compute λ for 49 β values which are evenly spaced in the β-interval.

The slight discrepancy in the two algorithms seen in figure 2.1.1 probably results from only using 49 grid points in the β-interval. A larger number of grid points both slows runtimes and can cause program errors. A solution to this would be to use an

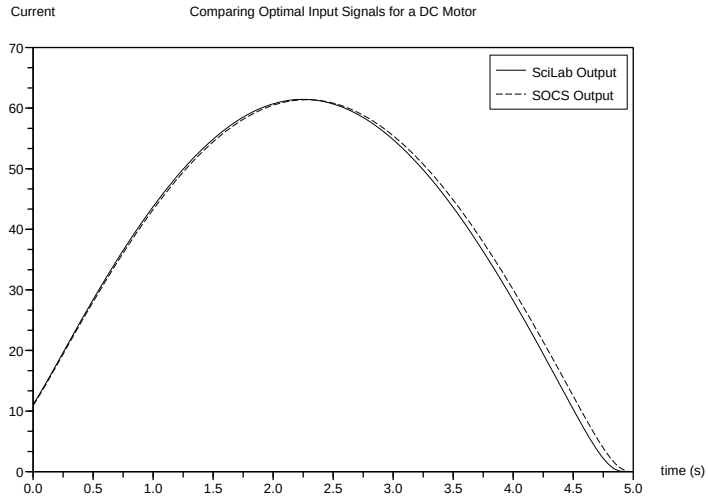


Figure 2.1: Comparison of the solution to the minimum control for fault detection of a DC motor using DO with SOCS and MEDS with SciLab.

iterative scheme that finds an approximate β^* over the entire interval, then redefines the β-interval to $[\beta^* - 0.1, \beta^* + 0.1]$ (where 0.1 is arbitrarily chosen) and rerun the algorithm. This can be continued until some desired precision is achieved. However, if we use the same number of grid points for each iteration, each added iteration will double the runtime. For larger systems this may be impractical, especially since the single run of the algorithm produces a result very close to the SOCS output.

2.1.2 Gas Turbine

M-1 tanks, helicopters, airplanes and ships all use gas turbines to generate power. One of the main advantages of the gas turbine is the power to weight ratio compared to a diesel engine [16]. A turbine has three main parts: the compressor, the combustion area and the turbine [16]. The compressor compresses incoming air to a high pressure. This pressurized air then passes into the combustion area, moving sometimes at hundreds of miles per hour. Inside the combustion area, a ring of fuel injectors releases a stream of fuel (typically kerosene, jet fuel, propane or natural gas). Fuel is burned to produce high pressure, high velocity gas, which in turns causes the

turbine to spin. The turbine, compressor, and the shaft connecting the turbine to the compressor all turn as a single unit. What the turbine does with the exhaust gas is dependent on the purpose of the machine [16].

We use a modified version of the 4th order, linearized gas turbine model from [17]. The states are high pressure speed x_1, low pressure speed x_2, nozzle area x_3, and fuel flow rate x_4 measured in bars, inches^2, kg/sec, and seconds, respectively. Our signal v is the referenced fuel flow rate[17].

Our model is in the form of (1.12) with

$$A_0 = \begin{bmatrix} -1.268 & -.04528 & 1.498 & 951.5 \\ 1.002 & -1.957 & 8.52 & 1240 \\ 0 & 0 & -10 & 0 \\ 0 & 0 & 0 & -100 \end{bmatrix}, B_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 100 \end{bmatrix} \tag{2.3a}$$

$$C_0 = \begin{bmatrix} I_{2\times 2}, 0_{2\times 2} \end{bmatrix}, N = \begin{bmatrix} 0_{2\times 4}, I_{2\times 2} \end{bmatrix}, M_0 = \begin{bmatrix} I_{4\times 4}, 0_{4\times 2} \end{bmatrix} \tag{2.3b}$$

We will assume there is a weakening in the turbine and that the fuel flow rate is reduced. The faulty model reflects this weakening and is manifested by a reduction by a factor of 15% on the 4th column of the matrix. Everything in equation (2.1.2) remains the same, except

$$A_1 = \begin{bmatrix} -1.268 & -.04528 & 1.498 & .808775 \\ 1.002 & -1.957 & 8.52 & 1.054 \\ 0 & 0 & -10 & 0 \\ 0 & 0 & 0 & -85 \end{bmatrix}.$$

This type of fault represents an important application of this algorithm. Not only does it allow the user to detect total system failures or near failures, but it also allows the user to detect weakenings of the system. Thus, the algorithm makes an excellent component of a condition-based maintenance system. Detecting a weakening before there is a catastrophic failure prevents long system downtimes, as well as reduces risks and costs associated with a total system loss.

We encountered several problems when we applied the algorithms to these models. Part of the problem was that the condition number of the state coefficient matrix is of

the order of 10^4. In order to alleviate this problem, we used a change of coordinates in the fuel flow rate. If we let $x = Qz$, where $Q = \text{diag}(1, 1, 1, .001)$, then (2.3) becomes $\dot{z} = Q^{-1}AQz + Q^{-1}Bv + Q^{-1}M\mu$ where

$$Q^{-1}AQ = \begin{bmatrix} -1.268 & -.04528 & 1.498 & .9515 \\ 1.002 & -1.957 & 8.52 & 1.240 \\ 0 & 0 & -10 & 0 \\ 0 & 0 & 0 & -100 \end{bmatrix}$$

.

$$Q^{-1}B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 100000 \end{bmatrix}, Q^{-1}M = \left[Diag(1, 1, 1, 1000), 0_{4\times 2} \right].$$

C does not change. This process reduces the condition number of A to 99.9. We will use a test period of $[0, 0.6]$. This test is performed on a short interval because in our faulty model, we changed the eigenvalue of largest magnitude. This results in a very rapid change in our model which requires a quick detection.

MEDS Algorithm

Once the model coordinates were changed to reduce the condition number of the state coefficient matrix, we were able to successfully find the auxiliary signal with the CC algorithm with a β grid size of $n = 50$. In initial runs of the code, the ODE solver would reach its maximum number of time steps before a solution was found. We believe this problem arose due to the stiffness of the model. Increasing the number of allowed time steps from 500 to 2000 enabled the code to converge successfully. The plot of λ versus β is smooth and has the standard inverted hyperbola shape. Our optimal β is $\beta^* = .4547$ and the optimal λ is $\lambda^* = 1.2329$. The optimal signal v^* has a norm of .8239 and is plotted in figure 2.1.2.

Direct Optimization

As with the MEDS algorithm, when using DO we ran into problems we believe were related to the condition number of the state coefficient matrix and overall stiff-

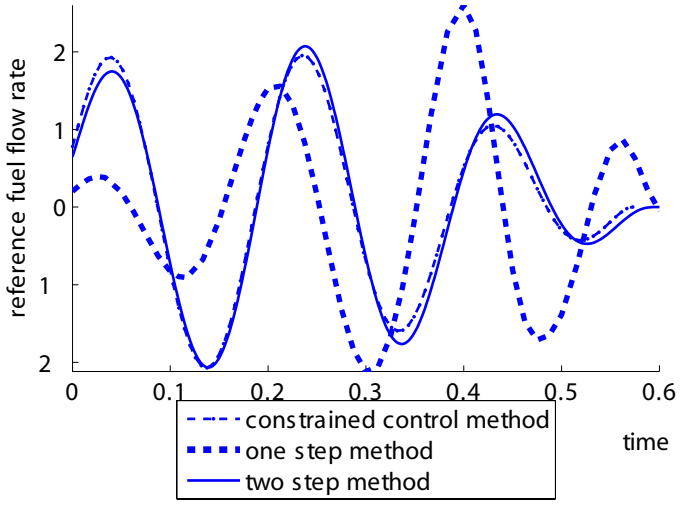


Figure 2.2: Optimal Signal for the gas turbine model using the MEDS and the DO algorithms.

ness of the turbine model. SOCS is a direct transcription code. Its first step after discretizing the problem is to find a feasible point. At first it was unable to find a feasible point. Changing coordinates alleviated the feasible point problem.

Then, we had trouble getting convergence. Our code stopped running after 3 grid iterations. In order to get convergence, we made two changes. First, we took the partial solution from the previous run, the solution on the third iteration of our non-convergent run, and used it as the initial guess of another run. SOCS first uses a lower

order trapezoidal method to solve the problem and then switches to a higher order Simpson method. We speculated that with a 'good' initial guess running through the trapezoidal part of SOCS's run, we would be more likely to get convergence. This strategy also had the effect of delaying the switch to Hermite Simpson.

We also noticed that the objective on SOCS's non-convergent run was only around 0.7. Previous experience with SOCS has shown that scaling the cost function will sometimes help. Using both the tailored initial guess and the weight 10^3 on the objective, SOCS converged at the desired error tolerance.

We also ran SOCS, using SOCS's default choice for an initial guess but with a weight on the objective. SOCS ran and converged. What was especially interesting in these runs is that the two signals produced by SOCS are very similar but not the same. The set boundary of the set of proper test signals in the additive uncertainty case form a surface which one can think of as an ellipsoid centered at the origin. The minimal norm proper v lie near the shortest axis of the ellipsoid. Since the surface is flattest at this point the norm of v is less sensitive to perturbations. This tells us that there are many suitable, or sub-optimal, signals which can be used. Our results tell us that either of these signals produced by SOCS is suitable. This is very positive since it could allow for a measure of approximation in the application of the signal.

The norm of the optimal signal using both a weighted cost and a tailored initial guess is 0.9044335 and the norm of the optimal signal using only the weighted cost is 0.9011104. Figure 2.1.2 shows both of these signals. The one step method refers to the weighted cost only output and the two step method refers to the weighted cost and initial guess output. The graphs in figure 2.1.2 are very similar. Notice that the signal tapers off to zero. This is the expected behavior of the signal. Although the three signals are similar, there are some differences. Also, the SOCS signals have a slightly larger norm than the signal from the MEDS algorithm. However, the MEDS and the SOCS two step are very close.

2.2 Conclusion

In this chapter we have shown how two model-based fault detection algorithms can be applied to power systems. We have also given some lessons learned in trouble shooting stiff problems. Finally, we have show that it is possible to have differences in the signals produced, depending upon how one implements the problem in software. These differences raise an important issue. Although there exists only one optimal signal, there can be several "almost optimal" signals. Any of these non-optimal signals can be used as the test signal since the algorithms which generate them use approximations to find the optimal signal.

Chapter 3

Suggestive Case Studies

In this chapter, we present several case studies in which we investigated the idea of using linearizations to find faults in nonlinear systems. Before we started any analysis, we wanted to ensure that this idea was feasible. We will use the same motor model for both case studies, however, we treat the faulty equation differently in each. For the first case study, we do the standard linearization and find the noise bound which holds. For the second case study, we treat the fault as an incipient fault, or a drift in the parameters. Finally, we compare the results of these two approaches. Computations involving SOCS were carried out by Ivan Andjelkovic.

3.1 Auxiliary Signal Design Around Set Points

We suppose that the faulty and unfaulty models of the nonlinear system are of the form

$$\dot{z}_i = f_i(z_i, u) + M_i\mu_i \tag{3.1a}$$

$$y = g_i(z_i, u) + N_i\mu_i. \tag{3.1b}$$

We assume the unfaulty system is operating near a set point $\bar{z}_0, \bar{u}$. That is, $0 = f_0(\bar{z}_0, \bar{u}, t)$ for all t.

There are scenarios where the faulty system $i = 1$ would have the same set point. Examples are sensor fault or faults of unused actuators. An example would be the

buildup of corrosion in valves only used infrequently. We consider a somewhat more general case here. We assume that the same set point control from the unfaulty model will produce a set point in the faulty model. But we do not require the set points to be the same. That is, there is a $\bar{z}_1$ such that $f_1(\bar{z}_1, \bar{u}, t) = 0$. The case where the faulty system does not have a set point is probably easier since there will be even greater difference in the dynamics between the two situations but we do not consider that case here.

Let $x_i = z_i - \bar{z}_i,\ v = u - \bar{u}$. Then the linearized models around the unfaulty and faulty set points are

$$\dot{x}_i = A_i x_i + B_i v + M_i \mu_i \tag{3.2a}$$

$$y = C_i x_i + D_i v + r_i + N_i \mu_i \tag{3.2b}$$

where $A_i = \frac{\partial f_i}{\partial z_i}(\bar{z}_i, \bar{u})$, $B_i = \frac{\partial f_i}{\partial u}(\bar{z}_i, \bar{u})$, $C_i = \frac{\partial g_i}{\partial z_i}(\bar{z}_i, \bar{u})$, $D_i = \frac{\partial g_i}{\partial u}(\bar{z}_i, \bar{u})$, $r_i = g_i(\bar{z}_i, \bar{u})$.

If $r_0 = r_1$, then we can define a new output variable as $y - r_0$ and apply the approach of [10] and compute an optimal test signal. If $r_0 - r_1 \neq 0$, then let $\tilde{y} = y - r_0$, $\Delta r = r_1 - r_0$ so that

$$\dot{x}_0 = A_0 x_0 + B_0 v + M_0 \mu_0 \tag{3.3a}$$

$$\tilde{y} = C_0 x_0 + D_0 v + N_0 \mu_0 \tag{3.3b}$$

$$\dot{x}_1 = A_1 x_1 + B_1 v + M_1 \mu_1 \tag{3.3c}$$

$$\tilde{y} = C_1 x_1 + D_1 v + \Delta r + N_1 \mu_1. \tag{3.3d}$$

If Δr is small relative to the uncertainty in the output equation, then it can be ignored and we proceed as in the $\Delta r = 0$ case. If Δr is not small, then (3.3) is an example where there is an additional input and the test signal may be found as in [18]. In the example of this section we will treat Δr as negligible and examine the effect of ignoring it.

3.1.1 Evaluation of the Test Signal

Given a test signal v we want to examine how good it really is on the nonlinear problem. To do this we will apply v to the original models and find the value of

the noise bound that actually holds. Note that the optimization problem includes a max norm inside it so that it is important that the optimization software not require necessary conditions. We will use the Sparse Optimal Control Software (SOCS) developed by the Boeing Corporation [13].

Our problem is to find $\min \max_{i=0,1} \{\mathcal{S}_0, \mathcal{S}_1\}$ where the noises produce a common output. This can be formulated as follows. Given a test signal v we minimize the parameter K whose minimum value will give the noise bound for which v is proper for the nonlinear system. That is,

$$\min K \tag{3.4a}$$

such that

$$\begin{aligned}
\dot{z}_0 &= f_0(z_0, v + \bar{u}) + M_0\nu_0 && (3.4b)\\
\dot{z}_1 &= f_1(z_1, v + \bar{u}) + M_1\nu_1 && (3.4c)\\
\dot{w}_0 &= \nu_0^T\nu_0,\ w_0(0) = 0 && (3.4d)\\
\dot{w}_0 &= \nu_1^T\nu_1,\ w_1(0) = 0 && (3.4e)\\
0 &= g_0(z_0, v + \bar{u}) - g_1(z_1, v + \bar{u}) + N_0\nu_0 - N_1\nu_1 && (3.4f)\\
K &\geq (z_0(0) - \bar{z}_0)^T P_0(z_0(0) - \bar{z}_0) + w_0(\tau) && (3.4g)\\
K &\geq (z_1(0) - \bar{z}_1)^T P_1(z_1(0) - \bar{z}_1) + w_1(\tau). && (3.4h)
\end{aligned}$$

3.1.2 Nonlinear Case Study

For our nonlinear case study we use a motor model from [19]. Let $x_1 = i_d$ and $x_3 = i_q$, where i_d and i_q are the $d-q$ axis currents and let $x_2 = \omega$ be the motor speed. Consistent with [19], we consider the motor speed to be hard to measure. Then the state dynamics from [19] along with our output equation and the addition of additive

uncertainty are modelled by

$$\begin{aligned}
\dot{x}_1 &= -\frac{R}{L}x_1 + Px_2x_3 + \frac{1}{L}v_d + \mu_1 && \text{(3.5a)}\\
\dot{x}_2 &= \frac{3P\phi_f}{2J}x_3 - \frac{B}{J}x_2 - \frac{T_L}{J} + 0.1\mu_2 && \text{(3.5b)}\\
\dot{x}_3 &= -\frac{R}{L}x_3 - Px_2x_1 - \frac{P\phi_f}{L}x_2 + \frac{1}{L}v_q + \mu_3 && \text{(3.5c)}\\
y &= x_1 + x_3 + \mu_4. && \text{(3.5d)}
\end{aligned}$$

Our control is $u = \begin{bmatrix} v_d \\ v_q \end{bmatrix}$, the $d-q$ axis control voltages. The design parameters are the stator resistance R, the stator inductance L, the number of pole pairs P, the permanent magnet flux ϕ_f, the rotor moment of inertia J, the viscous friction coefficient B, and the load torque T_L. The following values for the parameters which we use for nonlinear Model 0 are from [19]: $R = 17.201\ \Omega$, $L = 0.038136\ H$, $P = 4$, $J = 0.001197\ \text{kgm}^2$, $B = .00025\ \text{Nm/rad}/s$, $\phi_f = 0.052325\ V/\text{rad}/s$.

To simplify the discussion we note that (3.5) is in the form

$$\begin{aligned}
\dot{x}_1 &= -ax_1 + px_2x_3 + hv_1 + \mu_1 && \text{(3.6a)}\\
\dot{x}_2 &= bx_3 - cx_2 - T + 0.1\mu_2 && \text{(3.6b)}\\
\dot{x}_3 &= -ax_3 - px_2x_1 - dx_2 + hv_2 + \mu_3 && \text{(3.6c)}\\
y &= x_1 + x_3 + \mu_4 && \text{(3.6d)}
\end{aligned}$$

where the constants have been changed to make formula manipulation simpler.

It seems reasonable to consider what happens for different loads T on the motor. All constants are positive. Note also that these variables are transformations of the usual voltages and currents.

The Jacobian is

$$\mathcal{J} = \begin{bmatrix} -a & px_3 & px_2 \\ 0 & -c & b \\ -px_2 & -d - px_1 & -a \end{bmatrix}. \qquad \text{(3.7)}$$

The noinput case

If $v_0 = v_1 = T = 0$, then the origin is an equilibrium. The eigenvalues $\sigma(\mathcal{J})$ of $\mathcal{J}$ are $\{-a, \frac{-(c+a)\pm\sqrt{(a+c)^2-4bd}}{2}\}$. The nonlinear system is stable if the constants

are positive and nonzero. In fact, the nonlinear system is globally stable, since $V = \frac{1}{2}(x_1^2 + x_3^2 + \frac{d}{b}x_2^2)$ is a global Lyapunov function.

The Set Points

We consider faults that correspond to a change in parameter values. The uncertainty in the second equation of the model may be interpreted as uncertainty in the torque load during the test.

We consider different loads and compute set points about them. The set point equations are

$$0 = -ax_1 + px_2x_3 + h\overline{v}_1 \tag{3.8a}$$

$$0 = bx_3 - cx_2 - T \tag{3.8b}$$

$$0 = -ax_3 - px_2x_1 - dx_2 + h\overline{v}_2 \tag{3.8c}$$

where $\overline{v}_i$ are constants.

Properly working model set point

According to [19], the case $x_1 = 0$ is of special interest in terms of perfomance. In general, given $\bar{x}_1$ the set point is

$$\begin{bmatrix} \overline{x}_2 \\ \overline{x}_3 \end{bmatrix} = \begin{bmatrix} -c & b \\ -d - p\bar{x}_1 & -a \end{bmatrix}^{-1} \begin{bmatrix} T \\ -h\overline{v}_2 \end{bmatrix} \tag{3.9a}$$

$$\overline{v}_1 = -\frac{p}{h}\overline{x}_2\overline{x}_3 + \frac{1}{h}a\bar{x}_1 \tag{3.9b}$$

and the set points are parameterized by $\bar{x}_1, T, \bar{v}_2$.

Faulty model set point

In this study we take the faulty models to be given by a change in the parameters. To get the faulty model set point we use the same values of $T, \bar{v}_1, \bar{v}_2$ and solve (3.8) with the faulty parameter values to get the $\bar{x}_i$ for model 1.

Test Cases

We now consider a number of different set points corresponding to different loads. We take the faulty model to be a change in the viscous friction coefficient from B to $100B$. We take the test interval to be $[0, 1]$. Not all significant digits are shown in the computational results presented. We calibrated the bound computation by checking it on each pair of linear models. We got a bound of 0.999 to 0.996 which is quite close to the theoretical value of 1.

Test 1

We consider the at-rest configuration. Then $T_L = 0$, and

$$\bar{u} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \bar{z}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \bar{z}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \ \Delta r = 0,$$

$$A_0 = \begin{bmatrix} -451.044 & 0 & 0 \\ 0 & -.209 & 262.281 \\ 0 & -5.488 & -451.044 \end{bmatrix},$$

$$A_1 = \begin{bmatrix} -451.044 & 0 & 0 \\ 0 & -20.886 & 262.281 \\ 0 & -5.488 & -451.044 \end{bmatrix},$$

$$B = \begin{bmatrix} 26.2223 & 0 \\ 0 & 0 \\ 0 & 26.222 \end{bmatrix}, \ C = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix},$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \ N = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}.$$

Looking at the A_i matrices we see that they have large negative eigenvalues since $\sigma(A_0) = \{-451.04, -447.83, -3.42\}$ and $\sigma(A_1) = \{-451.04, -447.67, -24.26\}$. The norm of the test signal is 54.2. The v_1 signal was zero. This was to be expected since

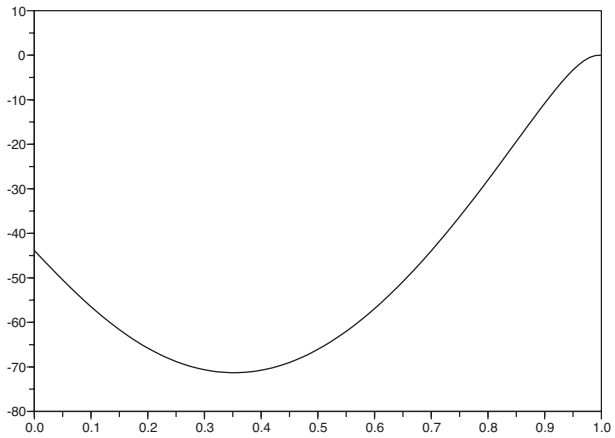


Figure 3.1: v_2 for Test 1.

for this set point the effect of v_1 was decoupled in the linearization from where the fault occurred. The v_2 signal is given in figure 3.1. The shape of the test signal is typical for a nonoscillatory system.

The error bounds from the nonlinear problem using (3.4) were computed using SOCS to be 0.708. Thus, in this example, the linear test signal computed with a noise bound of 1 would work well on the nonlinear problem but the signal would provide perfect detection only with the smaller additive noise bound of 0.708 on the nonlinear problem. For readers interested in using the algorithms from [10] we note that in the interests of speeding up the computation that a prelminary computation showed that β was close to 0.5 where the $\max_{\beta} \gamma_\beta$ occurred. Accordingly we restricted the β search to the interval $[0.45, 0.55]$. The modestly large size of the test signal suggested setting the γ iteration tolerance `prec` to 0.001 instead of the default 0.01.

Test 2

This is a small load case. We take $T = 1$, $\bar{v}_2 = 1.1$. Then

$$\bar{u} = \begin{bmatrix} -0.0053 \\ 1.1 \end{bmatrix}, \bar{z}_0 = \begin{bmatrix} 0 \\ 4.63870 \\ 0.00751 \end{bmatrix},$$

$$\bar{z}_1 = \begin{bmatrix} 1.6414e-005 \\ 0.6551 \\ 0.0560 \end{bmatrix}, \Delta r = .0485.$$

Note that $\sigma(A_0) = \{-449.44 \pm 18.54i, -3.4199\}$, $\sigma(A_1) = \{-449.36 \pm 1.927i, -24.26\}$. The v_1 test signal is in figure 3.2. v_2 is similar to figure 3.1. The norm of v is 54.84. This is about the same as in Test 1. The error bounds from the nonlinear problem are

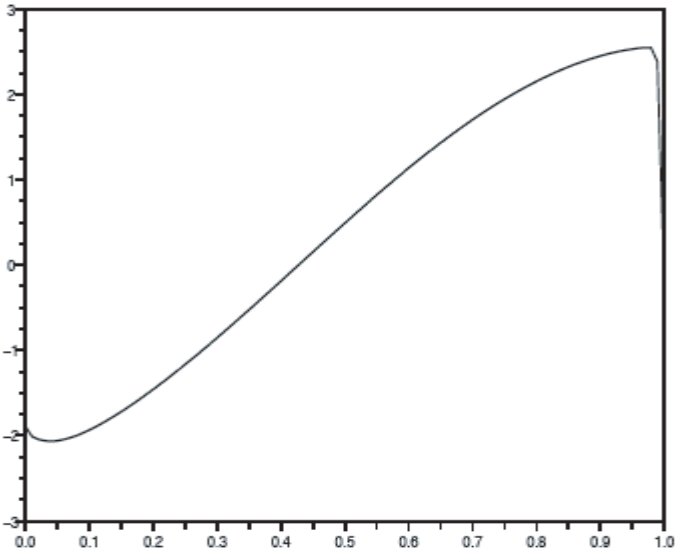


Figure 3.2: v_1 for Test 2.

computed to be 1.063. On this problem, the v was proper on the nonlinear problem for essentially the same noise bounds as for the linearizations.

Test 3

This is a larger load and set point control action case. We take $T = 10$, $\bar{v}_2 = 100$. Then

$$\bar{u} = \begin{bmatrix} -26.699 \\ 100 \end{bmatrix}, \bar{z}_0 = \begin{bmatrix} 0 \\ 445.495 \\ 0.393 \end{bmatrix}$$

$$\bar{z}_1 = \begin{bmatrix} 0.853 \\ 58.124 \\ 4.667 \end{bmatrix}, \Delta r = 5.13.$$

Note that $\sigma(A_0) = \{-450.84 \pm 1782.33i, -0.62\}$, $\sigma(A_1) = \{-446.46 \pm 229.14i, -30.06\}$. Near the set point there is a highly oscillatory component. This problem also had the largest Δr. The test signal v_1 is given in figure 3.3. v_2 is similar in size and shape. For this problem with the default settings the numerical code returned a test signal on $[0, .37]$ instead of $[0, 1]$. By greatly tightening a number of tolerances we got a signal on $[0, 0.55]$ but with essentially the same norm. Viewed as a function of the length of the test period the norm of the test signal drops and then approaches a nonzero value. In this case, the norm has essentially leveled off by 0.38 and our software detects that nothing is to be gained from using a longer test period. The norm of v is 31.6 which

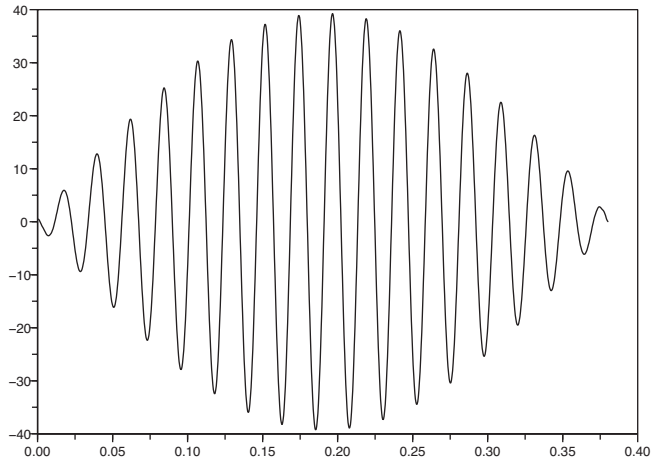


Figure 3.3: v_1 for Test 3.

is smaller than in the other test cases. The error bound calculation which worked so well on Tests 1, 2, 4 had trouble converging on this problem. The cause of this is being investigated.

Test 4

This is a heavier load but less control action than Test 3. We take $T = 50$, $\bar{v}_2 = 10$. Then

$$\bar{u} = \begin{bmatrix} -0.987 \\ 10 \end{bmatrix}, \bar{z}_0 = \begin{bmatrix} 0 \\ 30.1389 \\ 0.2146 \end{bmatrix},$$

$$\bar{z}_1 = \begin{bmatrix} -0.0373 \\ 4.2717 \\ 0.5308 \end{bmatrix}, \Delta r = 0.28.$$

Note that $\sigma(A_0) = \{-449.48 \pm 120.71i, -3.33\}$, $\sigma(A_1) = -449.38 \pm 16.41i, -24.21\}$.

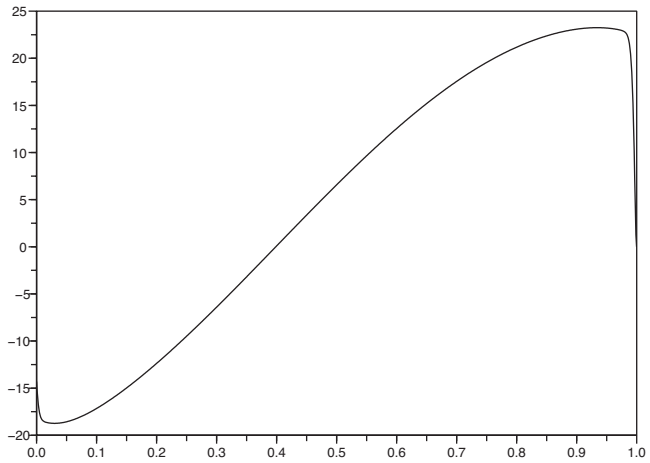


Figure 3.4: v_1 for Test 4.

The oscillatory component is less in this model but the problem is still a stiff one. The test signal is given in figures 3.4 and 3.5. The norm of v is 57.7. The error bounds from the nonlinear probem are 1.117.

The rapid changes at the ends of the test signals in figures 3.4 and 3.5 are not graphical artifacts but rather are boundary layers since the problem defining the test signal has two time scales. The boundary layers are not important as far as using v since they have small L^1 norm and hence we can just ignore them and get a v which will work as well. However, they are important when evaluating or computing a test

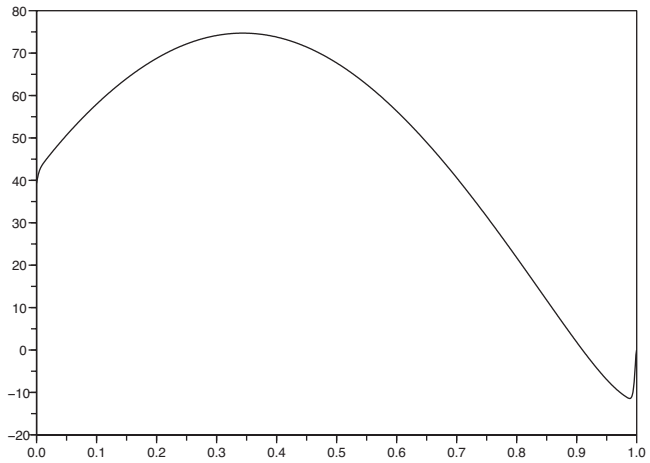


Figure 3.5: v_2 for Test 4.

signal since the boundary effects will cause the optimizers and integrators to have to use finer grids.

3.1.3 Discussion of Results

The ignoring of Δr did not stop us from finding useful test signals. The shape of the test signal and its size varied with where the set point was. In the cases to date on this problem we saw that sometimes the test signal actually performed better on the nonlinear problem then it did on the linearized problem. This was especially true with higher loads.

3.1.4 Discussion on Software & Its Integration

Several software packages were used in the process of solving these problems including Scilab, MATLAB, Maple and SOCS. The first step was in determining the set points which required a nonlinear solve. We also wanted to look at the eigenvalues of the linearization. There are several ways to generate linearized models depending on how the original model is formulated. Both MATLAB and Scilab can generate linearizations as can MAPLE. Our nonlinear solve was done in MATLAB `http://matlab.com`. Once the models were determined, the test signals were found

for the linearizations. Two approaches could be used. One is to form the system matrices for the optimization problem which results in the test signal in as seen in Chapter 4 of [10]. For this problem we found the detection signal for the linearized problem using Scilab coded algorithms from Chapter 3 of [10]. Scilab is a MATLAB-like software environment developed at INRIA [20, 21]. Scilab is freely available and may be downloaded from `http://www.scilab.org/`. While we used Scilab, the problem could easily have been solved in MATLAB. A port from Scilab to MATLAB is complete. Due to the stiffness of the model with which we were working, solving the problem was somewhat challenging for the software. In the higher load cases we needed to alter the defaults in the integrators and substantially increase the allowed number of iterations and time steps.

Finding the error bounds for the nonlinear problem required a more sophisticated optimization package which is able to solve the optimization problem without being given necessary conditions. We used Sparse Optimal Control Software (SOCS) produced by The Boeing Company. SOCS is a powerful package of FORTRAN subroutines able to solve very complex problems. More information about SOCS is available at `http://www.boeing.com/phantom/socs/ index.html`. On the higher load problems we needed to take finer initial grids than the default and also to make large initial guesses for K to help with finding a feasible solution.

3.1.5 Conclusions

We have examined the use of test signals found from linearized models on a nonlinear problem. The test signals worked well and often were proper for higher noise bounds than in the original linear problem. We have shown how to evaluate these test signals on the nonlinear problem. The solution of these problems can require the integration of different solftware packages.

3.2 Auxiliary Signal Design for Incipient Faults

Next we consider the incipient fault algorithm of [22, 1, 23, 24]. Suppose we have the model of the form [22]:

$$\dot{x}_0 = A(\theta_0)x_0 + B(\theta_0)v + M\mu_0 \tag{3.10a}$$

$$y = C(\theta_0)x_0 + D(\theta_0)v + N\mu_0 \tag{3.10b}$$

where θ is a scalar parameter. The algorithm is designed to find a test signal to detect drift in that parameter. If the small drift is represented by some small change in the parameter, $\delta\theta$, we can say we have the two models:

$$\dot{x}_0 = A(\theta_0)x_0 + B(\theta_0)v + M\mu_0 \tag{3.11a}$$

$$\dot{x}_1 = A(\theta_0 + \delta\theta)x_1 + B(\theta_0 + \delta\theta)v + M\mu_1 \tag{3.11b}$$

$$y = C(\theta_0)x_0 + D(\theta_0)v + N\mu_0 \tag{3.11c}$$

$$y = C(\theta_0 + \delta\theta)x_1 + D(\theta_0 + \delta\theta)v + N\mu_1 \tag{3.11d}$$

where θ is a scalar parameter and $\delta\theta$ is taken to be some *unknown* drift. If $\delta\theta$ is known and fixed, then we could use the 2-model approach from the previous section. Here we prove differently.

If we let [22] $\eta = \begin{bmatrix} \tilde{x} \\ \bar{x} \end{bmatrix}$ where $\tilde{x} = x_1 - x_0$ and $\bar{x} = x_1\delta\theta$ and $w = v\delta\theta$, we can transform the system (3.11) into

$$\dot{\eta} = \begin{bmatrix} A(\theta_0) & \frac{\partial A}{\partial \theta}(\theta_0) \\ 0 & A(\theta_0) \end{bmatrix} \eta + \begin{bmatrix} \frac{\partial B}{\partial \theta}(\theta_0) \\ B(\theta_0) \end{bmatrix} w + \begin{bmatrix} -M & M \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_1 \end{bmatrix} \tag{3.12a}$$

$$0 = \begin{bmatrix} C(\theta_0) & \frac{\partial C}{\partial \theta}(\theta_0) \end{bmatrix} \eta + \frac{\partial D}{\partial \theta}(\theta_0)w + \begin{bmatrix} -N & N \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_1 \end{bmatrix} \tag{3.12b}$$

after eliminating the common y output from the system and dropping higher order terms from the approximation. Using this transformation, the models are in the same form as (1.21). D does not appear in our application and will be dropped from what follows.

Because the drift in the parameters is taken to be small, we will also we assume that

$$\bar{x}(0) = 0. \tag{3.13}$$

This point will be revisited later in this discussion.

There is a change in the noise model now that we have transformed the equations. This formulation simplifies the optimization problem. We have

$$\phi_\beta(w) = \inf_{\eta,\mu} \beta \mathcal{S}_w(x_0(0), \mu_0) + (1-\beta)\mathcal{S}_w(x_0(0), \mu_0),$$

subject to (3.12) and (3.13). As shown in [22], the max across β of (1.15) occurs when $\beta = \frac{1}{2}$. Moreover, at that max, the noise model satisfies $S(x_0(0), \mu_0) = S(x_1(0), \mu_1)$. This leads to Lemma 4.1 in [22], which is as follows:

Lemma 1 *For $\beta = \frac{1}{2}$, the optimization problem becomes*

$$\phi_{\frac{1}{2}}(w) = \inf_{\mu,\eta} \frac{1}{4}(\bar{x}(0)^T P_0^{-1} \bar{x}(0) + \|\mu\|^2$$

subject to the equations (3.12) and the boundary condition (3.13).

An important feature of this algorithm is that there is no choice of $\delta\theta$ used in the algorithm. The algorithm finds a signal $w = v\delta\theta$. Thus, in theory, the signal should be 'one size fits all'. Naturally, we can see that as $\delta\theta$ gets small, the size of $v = \frac{w}{\delta\theta}$ could grow very large. This is logical when one considers that the models would naturally be very close together in that case and require more energy to separate them.

As with the MEDS algorithm, we need to solve the optimization problem in Lemma 1. This can be done with either the CC l approach or the DO approach discussed below. In [22] the Constrained Control method is discussed.

In the paper [1] we used both the CC and the DO method for finding the detection signals. We found signals for the MEDS algorithm as it applies to two models using the CC algorithm. However, for the incipient algorithm, we used the DO approach. Since the DO solution to the incipient problem has not be discussed elsewhere, we include it here.

We use the model reduction found in [25] and described in Section 1.3.2. However, as we have a fundamentally different model, some small changes are made to the reduction. Assume that we have the models given by (3.12). The basic idea behind the DO approach is to reformulate the problem using the minimization across the states and noises as constraints on the optimization problem. Before doing this, however, we also reduce the order of the model. We know that there exists an orthogonal matrix Q such that $NQ = R^T$ $R^T = [\bar{N}\ 0]$ for some invertible $\bar{N}$. So, we can rewrite (3.12b) as

$$0 = \begin{bmatrix} C & C(\theta_0) \end{bmatrix} \eta + \begin{bmatrix} -R^T & R^T \end{bmatrix} \begin{bmatrix} Q^T\mu_0 \\ Q^T\mu_1 \end{bmatrix}.$$

And, if we let $\begin{bmatrix} Q^T\mu_0 \\ Q^T\mu_1 \end{bmatrix} = \begin{bmatrix} \bar{\bar{\mu}}_0 \\ \bar{\bar{\mu}}_1 \end{bmatrix}$, then we have

$$0 = \begin{bmatrix} C & C(\theta_0) \end{bmatrix} \eta + \begin{bmatrix} -R^T & R^T \end{bmatrix} \begin{bmatrix} \bar{\bar{\mu}}_0 \\ \bar{\bar{\mu}}_1 \end{bmatrix}.$$

This transforms the noise in (3.12b). We must transform the noise in (3.12a). Let $\bar{\bar{M}} = MQ$ so that $M\mu_i = MQQ^T\mu_i = \bar{\bar{M}}\bar{\bar{\mu}}_i$. Then, we can rewrite (3.12a) as

$$\dot{\eta} = \begin{bmatrix} A & A(\theta) \\ 0 & A(\theta) \end{bmatrix} \eta + \begin{bmatrix} B(\theta) \\ B \end{bmatrix} w + \begin{bmatrix} -\bar{\bar{M}} & \bar{\bar{M}} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{\bar{\mu}}_0 \\ \bar{\bar{\mu}}_1 \end{bmatrix}. \tag{3.14}$$

Let $\bar{\bar{\mu}}_i = \begin{bmatrix} \bar{\mu}_i \\ \hat{\mu}_i \end{bmatrix}$ with the appropriate dimensions so that

$$R^T\bar{\bar{\mu}}_i = \begin{bmatrix} \bar{N}\bar{\mu}_i \\ 0\hat{\mu}_i \end{bmatrix}.$$

Then we have $0 = C\tilde{x} + C(\theta)\bar{x} - \bar{N}\bar{\mu}_0 + \bar{N}\bar{\mu}_1$ where $\bar{N}$ is invertible. Using this equation, we can solve for one of the noises. Thus, we have $\bar{\mu}_0 = \bar{N}^{-1}C\tilde{x} + \bar{N}^{-1}C(\theta)\bar{x}$ $+\bar{N}^{-1}\bar{N}\bar{\mu}_1$ which is

$$\bar{\mu}_0 = \bar{N}^{-1}C\tilde{x} + \bar{N}^{-1}C(\theta)\bar{x} + \bar{\mu}_1.$$

Now we put this back in our first equation. Let

$$\bar{\bar{M}}\bar{\bar{\mu}}_i = \begin{bmatrix} \bar{M}\bar{\mu}_i \\ \hat{M}\hat{\mu}_i \end{bmatrix},$$

so that $M = \begin{bmatrix} -\bar{M} & -\hat{M} & \bar{M} & \hat{M} \\ 0 & 0 & 0 & 0 \end{bmatrix}$ and (3.14) is

$$\dot{\eta} = \begin{bmatrix} A & A(\theta) \\ 0 & A(\theta) \end{bmatrix} \eta + \begin{bmatrix} B(\theta) \\ B \end{bmatrix} w + \begin{bmatrix} -\bar{M} & -\hat{M} & \bar{M} & \hat{M} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{\mu}_0 \\ \hat{\mu}_0 \\ \bar{\mu}_1 \\ \hat{\mu}_1 \end{bmatrix}. \quad (3.15a)$$

Substituting for $\bar{\mu}_0$ we get

$$\dot{\eta} = \begin{bmatrix} A & A(\theta) \\ 0 & A(\theta) \end{bmatrix} \eta + \begin{bmatrix} B(\theta) \\ B \end{bmatrix} w + \begin{bmatrix} -\bar{M} & -\hat{M} & \bar{M} & \hat{M} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Sigma \\ \hat{\mu}_0 \\ \bar{\mu}_1 \\ \hat{\mu}_1 \end{bmatrix} \quad (3.16a)$$

where $\Sigma = \bar{N}^{-1}C\tilde{x} + \bar{N}^{-1}C(\theta)\bar{x} + \bar{\mu}_1$.

Multiplying this out and reforming our matrices, we get an equation in the form of $\dot{\eta} = A\eta + Bw + M\mu$ where B, η and w remain the same but

$$A = \begin{bmatrix} A - \bar{M}\bar{N}^{-1}C & A(\theta) - \hat{M}\bar{N}^{-1}C(\theta) \\ 0 & A \end{bmatrix},$$

$$M\mu = \begin{bmatrix} -\hat{M} & 0 & \hat{M} \\ 0 & 0 & 0 \end{bmatrix}, \ \mu = \begin{bmatrix} \hat{\mu}_0 \\ \bar{\mu}_1 \\ \hat{\mu}_1 \end{bmatrix}.$$

The second step in transforming the system is the new cost. Now, as we know that $\beta = \frac{1}{2}$, we have

$$\begin{aligned} \tfrac{1}{2}\int_0^T \|\mu\|^2 dt &= \tfrac{1}{2}\int_0^T \|\bar{\mu}_0\|^2 + \|\hat{\mu}_0\|^2 + \|\bar{\mu}_1\|^2 + \|\hat{\mu}_1\|^2 dt \\ &= \tfrac{1}{2}\int_0^T \|\bar{N}^{-1}C\tilde{x} + \bar{N}^{-1}C(\theta)\bar{x} + \bar{\mu}_1\|^2 + \|\hat{\mu}_0\|^2 + \|\bar{\mu}_1\|^2 + \|\hat{\mu}_1\|^2 dt. \quad (3.17) \end{aligned}$$

Multiplying this out, we get an equation in the form of $\frac{1}{2}\eta^T Q\eta + \frac{1}{2}\eta^T H\mu + \frac{1}{2}\mu^T R\mu$. where

$$Q = \begin{bmatrix} C^T C & C^T C(\theta) \\ C(\theta)^T C & C(\theta)^T C(\theta) \end{bmatrix},$$

$$H = \begin{bmatrix} 0 & 2C^T\bar{N}^{-1} & 0 \\ 0 & 2C(\theta)^T\bar{N}^{-1} & 0 \end{bmatrix},$$
$$R = \text{diag}(1, 2, 1).$$

The optimization problem that we will solve is

$$\min_v \int_0^\omega \parallel v \parallel^2 dt$$

subject to the constraints

$$\dot{\eta} = A\eta + Bv + M\mu \tag{3.19a}$$
$$\dot{\lambda} = -Qx - \frac{1}{2}H\mu - A^T\lambda,\ \lambda(T) = 0 \tag{3.19b}$$
$$\dot{Z} = \frac{1}{2}[\eta^T Q\eta + \eta^T H\mu + \mu^T R\mu] \tag{3.19c}$$
$$0 = R\mu + \frac{1}{2}H^T\eta + M^T\lambda \tag{3.19d}$$
$$Z(0) = \frac{1}{2}\eta_0^T P\eta_0,\ Z(T) \geq 1 \tag{3.19e}$$
$$\tilde{\lambda}(0) = -P_0\tilde{x} \tag{3.19f}$$
$$\bar{x}(0) = 0 \tag{3.19g}$$

As a brief technical note, the use of $\frac{1}{2}$ in the above equations might seem like it diverges from the theorem given. However, the $\frac{1}{2}$ in the above equations is the $\beta = \frac{1}{2}$. The other $\frac{1}{2}$ is usually included to create the same type of problem one might see in solving an LQR problem. It would simply have been absorbed into the matrices by including a 2 so we do not include it as the rest of our explanation is more straight forward without this contrived constant.

3.2.1 Case Study

In order to study the incipient algorithm, we consider a particular and challenging example of model of a motor [19]. As a reminder from the previous section, the model

is a 3 dimensional system, with several parameters, given by:

$$\begin{aligned}
\dot{x}_1 &= -\frac{R}{L}x_1 + Px_2x_3 + \frac{1}{L}v_d + \mu_1 && (3.20a)\\
\dot{x}_2 &= \frac{3P\phi_f}{2J}x_3 - \frac{B}{J}x_2 - \frac{T_L}{J} + 0.1\mu_2 && (3.20b)\\
\dot{x}_3 &= -\frac{R}{L}x_3 - Px_2x_1 - \frac{P\phi_f}{L}x_2 + \frac{1}{L}v_q + \mu_3 && (3.20c)\\
y &= x_1 + x_3 + \mu_4 && (3.20d)
\end{aligned}$$

The design parameters are R, the stator resistance, L, the stator inductance, P, the number of pole pairs, ϕ_f, the permanent magnet flux, J, the rotor moment of inertia, B, the viscous friction coefficient, and T_L, the load torque. The controls are v_d and v_q. The parameters for the model are given by R =17.201 Ω, L = 0.038136 H, P = 4, J = 0.001197 kgm^2, B = 0.00025 Nm/rad/s, and ϕ_f = 0.052325 V/rad/s.

We consider a linearization of this model. The application of the MEDS algorithm to the linearization is discussed at length in [26], as well as in the previous section. We linearize around an operating point of the nonlinear model. The process of linearizing requires one to choose values for the load torque, T_L, and values for the controls. In [26] and above, several values were considered and discussed. Here we will use those values and set points without further explanation. If $\hat{x}$ is an operating point, then the linearized model is given by

$$\begin{aligned}
\dot{x} &= Ax + Bv + M\mu && (3.21a)\\
y &= Cx + N\mu && (3.21b)
\end{aligned}$$

with

$$A = \begin{bmatrix} -\frac{R}{L} & P\hat{x}_3 & P\bar{x}_2 \\ 0 & -\frac{B}{J} & \frac{3P\phi_f}{2J} \\ -P\hat{x}_2 & -\frac{P\phi_f}{L} - P\hat{x}_1 & -\frac{R}{L} \end{bmatrix}, \; B = \begin{bmatrix} \frac{1}{L} & 0 \\ 0 & 0 \\ 0 & \frac{1}{L} \end{bmatrix}, \; M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ and $N = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$.

In this case study, we consider several different values of δB, including $\delta B = 0.1, 0.5, 4, 9, 99$. Although we will only discuss a few of the cases, most of the results were more similar than one might have expected given the disparity in the size of the parameter drifts considered.

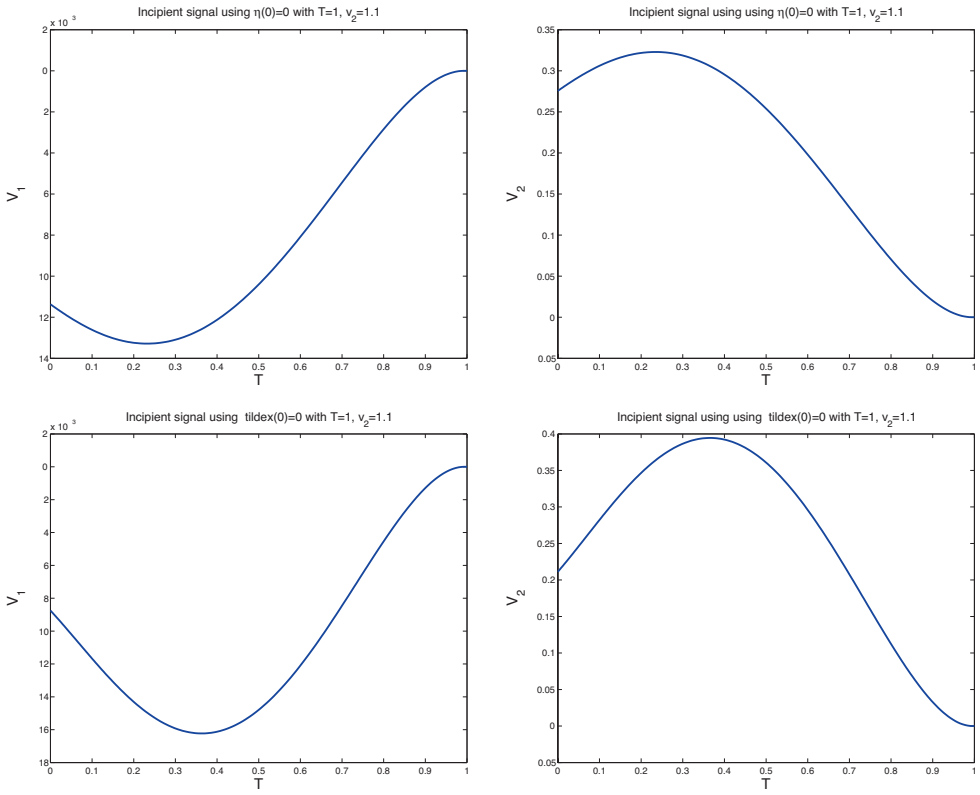


Figure 3.6: Comparison of signals using different boundary conditions.

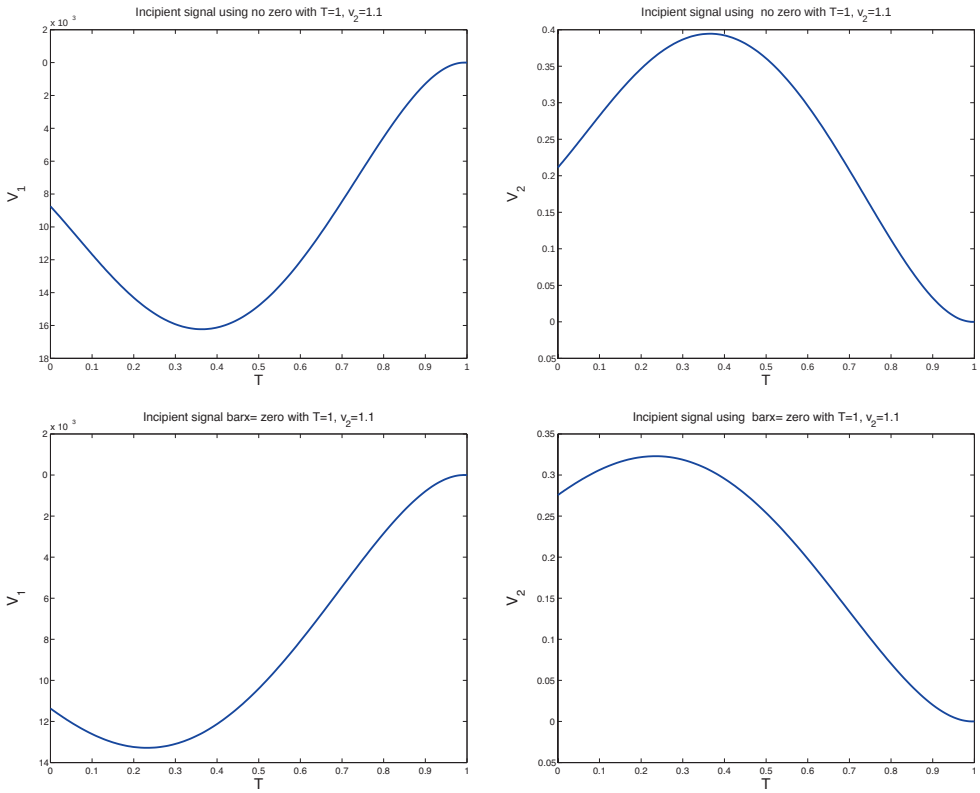


Figure 3.7: More comparisons of signals using different boundary conditions.

Boundary Condition

In the noise measure (1.13), the matrix P puts a weight on the uncertain initial condition. If $P = 0$, it means there is no information on the initial state of the system. The boundary condition of (3.13) presents an interesting question. One could argue that because δB is very small by assumption and thus $\bar{x} = x_1 \delta B$ will also be very small, that it makes sense to use the condition that $\bar{x}(0) = 0$. On the other hand, one could argue that in a stable system that has been operating prior to the test period, x_0 and x_1 will be very close together and thus $\tilde{x}(0) = x_0(0) - x_1(0)$ will be very small, leading to the condition that $\tilde{x}(0) = 0$. One could also say that both of these arguments are valid and both $\tilde{x}(0)$ and $\bar{x}(0)$ should be zero or that neither are valid and the weight P used in the MEDS algorithm is most appropriate.

In figures 3.6 and 3.7, we see the signal for the parameter values $T = 1$ and $v_2 = 1.1$ for the cases discussed above. The top set of signals is the case where $\eta(0) = 0$, followed by $\tilde{x}(0) = 0$, neither are 0 and finally, $\bar{x}(0) = 0$. We see that while the boundary conditions are different, they produce a similar result. We also see that while they all reach their highest point at approximately the same place in the test period, the case of having neither be 0 or having $\tilde{x}$ be 0 start smaller and require more of a push in the beginning. Naturally, the magnitude of the signals was similar in the pairs of cases that produced such similar signals. Note that v_1 is much smaller than v_2.

Comparison of MEDS algorithm signals and incipient algorithm signals

In this section, we compare the signals generated with the MEDS algorithm for a fixed perturbation with the signals generated by the incipient algorithm. In figure 3.8, we see the signals generated by both the incipient algorithm and the MEDS algorithm for the case where $T = 1$ and $v_2 = 1.1$. The top set of plots is the w generated by the incipient algorithm where as the bottom set of plots are the MEDS algorithm signals with $\delta B = 99B \approx 0.02$. The signals are very similar in shape. Dividing w by δB gives signals of comparable, but not identical size of the MEDS signal. It is also interesting to note that the MEDS signal is shaped more like those signals generated

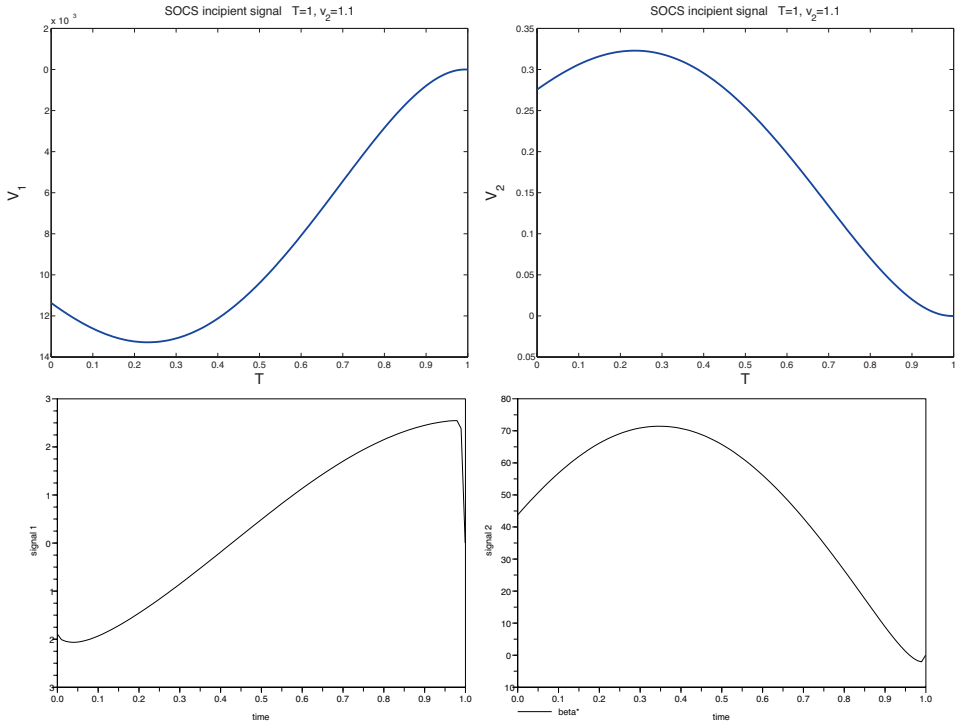


Figure 3.8: Signals from $\delta B = 99B, T = 1, v_2 = 1.1$.

by the incipient algorithm when used with the boundary condition of $\tilde{x} = 0$ or no extra boundary condition. The MEDS signal seems to start smaller and rise more quickly, as in those cases.

The incipient algorithm was designed thinking in terms of small changes in parameters. We consider the case when $\delta B = 0.1B$. This would be a very small change in the parameter as the starting value of $B = 0.00025$. Figure 3.9 shows the results of this case when the parameters were $T = 0, v_2 = 0$. Note that v_1 is numerically zero in the top left graph. The DO approach was used to find the MEDS signal, rather than the CC approach as with the other MEDS signals. The CC approach was unable to converge on a change this small. We did not entirely have convergence using the DO approach on this problem either. However, the errors in constraint satisfaction were sufficiently small for us to be able to use these signals. Error in the ODE was $1.5145E-05$ and error in the DAE was $1.5145E-05$.

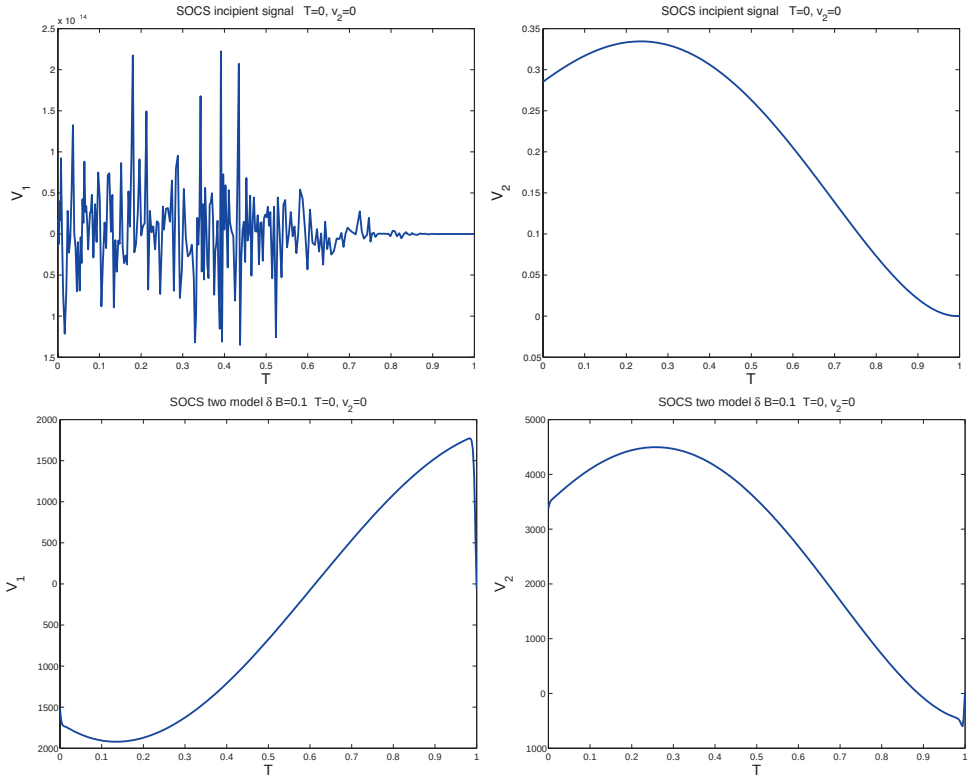


Figure 3.9: Signals from $\delta = .1B, T = 0, v_2 = 0$.

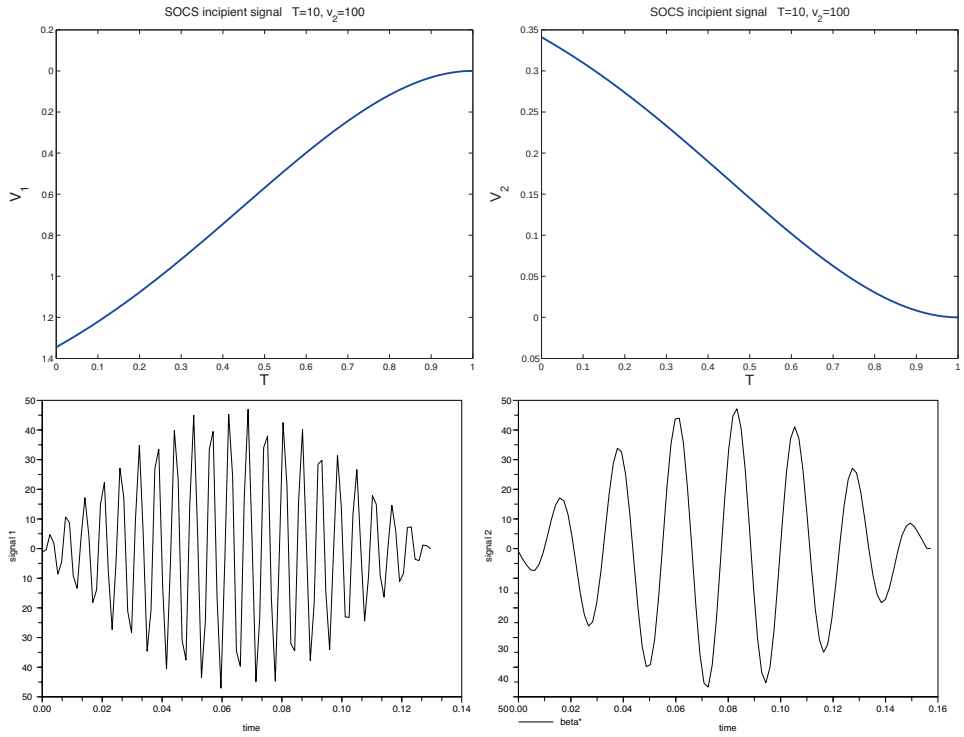


Figure 3.10: Signals from $\delta = 99B, T = 10, v_2 = 100$.

One of the reasons this motor model is of particular interest is because different sets of parameters bring out such different dynamic behavior in the signals and model. Unfortunately, the incipient algorithm was not able to capture this behavior. In figure 3.10 we see the signals generated by the incipient algorithm and the MEDS algorithm. The MEDS signal, at the bottom of the figure, is oscillating. The signal, w, generated by the incipient algorithm did not capture this oscillation at all. As in the previous case, the magnitude of the signal is nowhere near that of the MEDS signal.

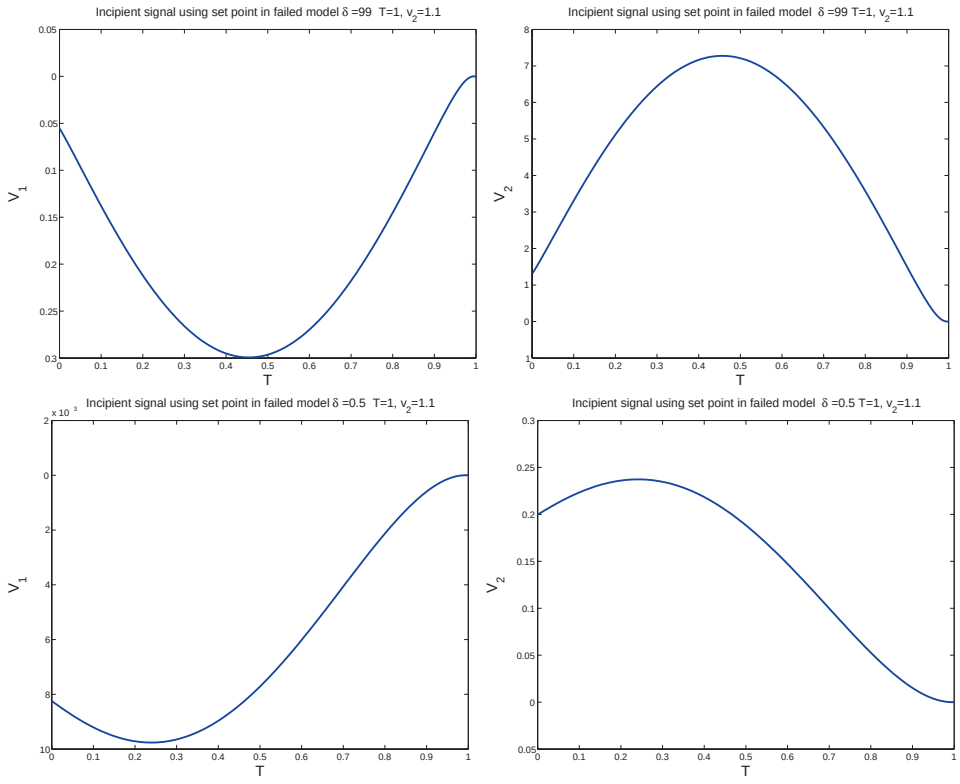


Figure 3.11: Signals from $T = 1, v_2 = 1.1$.

The nonlinear problem

We examined how the algorithms would do if the set point generated by the properly working model was used to linearize the failed model and then the incipient algorithm was applied to this new model. Figure 3.11 shows the results of this test.

They are very similar to the results of our other tests. The top set of plots in the figure is the case where $T = 1, v_2 = 1.1$ were used to find the set points.

3.2.2 Conclusions

We have presented the first implementation of an active approach to determine incipient faults using test signals. The test signal is computed off line and then scaled depending on the desired detection level. We have used a challenging nonlinear example as the test problem. The signals from the new algorithm were compared, at different fault levels, with the optimal test signal for that fault level. The test signals from the incipient algorithm were seen to work well even when there were large changes in the faulty parameters. The approach appears to be practical and merits continued development.

Chapter 4

Nonlinear Analysis

In the previous chapter, we showed that linearizations are a reasonable approach to find faults in nonlinear systems. In this chapter we present analysis to prove that this is in fact the case. In the first section, however, we give some background information that is needed for our analysis. The next two sections give the results for when optimal signals from linearizations will detect faults in the nonlinear system.

4.1 Nonlinearities

In this chapter our approach to solving the nonlinear problem involves the use of linearizations. We will assume that both our working and faulty systems are only nonlinear in the states, i.e. both are of the form

$$\dot{x} = g(x) + Bv + M\mu. \tag{4.1}$$

Also, we will require that $g(x)$ is locally Lipschitz, $g(0) = 0$, and we will linearize around $\bar{x} = 0$. Thus, we have $g(x) = g'(\bar{x})(x - \bar{x}) + O(\|x\|^2) = g'(0)x + O(\|x\|^2) = Ax + O(\|x\|^2)$. Dropping the $O(\|x\|^2)$ term, we get the approximation

$$\dot{x} = Ax + Bv + M\mu. \tag{4.2}$$

A natural thing to try on a nonlinear system is to use a test signal developed from the linearization. This is not always possible as the next simple example shows.

Example. Consider the simple nonlinear system

$$\dot{x}_0 = -\frac{9}{2}x_0 + x_0^2 + 2v \tag{4.3a}$$
$$y_0 = x_0 + \mu_0 \tag{4.3b}$$

and the faulty nonlinear system

$$\dot{x}_1 = -\frac{5}{2}x_1 + x_1^2 + \frac{3}{2}v \tag{4.4a}$$
$$y_1 = x_1 + \mu_1. \tag{4.4b}$$

The linearizations are

$$\dot{x}_0 = -\frac{9}{2}x_0 + 2v \tag{4.5a}$$
$$y_0 = x_0 + \mu_0 \tag{4.5b}$$

and

$$\dot{x}_1 = -\frac{5}{2}x_1 + \frac{3}{2}v \tag{4.6a}$$
$$y_1 = x_1 + \mu_1, \tag{4.6b}$$

respectively. Suppose in (1.13) that $Q_i = 1$, $\gamma = 0.251$, and $\omega = 100$. Suppose that we consider the test signal $v = 1$. The noise bound implies that $|x_i(0)| < \frac{1}{2}$. Solving (4.5a) and (4.6a) for x_1, x_2 and setting $y_0 = y_1$ using (4.5b) and (4.6b), it is then easy to show that $\mu_1 - \mu_0$ must violate the noise bound (1.13). Thus, this v is strictly proper for the linearized system. But if we set $v = 1$ in the nonlinear systems (4.3a) and (4.4a), we see that $x_0 = x_1 = \frac{1}{2}$ satisfies (4.3),(4.4) and $y_0 = y_1$ with $\mu_0 = \mu_1 = 0$. Thus the original linear test signal v does not work for the nonlinear system for the given noise bounds. On the other hand, it is also possible for the nonlinear problem to have a proper v but the linearization may not. For example, two different models could have the same linearization [27].

4.2 Guaranteed Fault Detection Through Scaled Linearizations

We are examining when we can guarantee fault detection in the nonlinear problem using a test signal from the linearization. Example 4.1 shows this is not always possible without extra assumptions or by changing what is sought. The idea behind our first result is to find the proper v for the linearized problem and then to scale this v, as well as μ and x_0, by a scalar β until we find a proper signal for the scaled nonlinear problem. To see what we mean by "scaling" take equation (1.12) and multiply it by β and equation (1.13) by β^2. Note that we multiply the noise by β^2 since this is actually a squared norm. Thus, we have

$$\begin{aligned} \beta\dot{x} &= A\beta x + B\beta v + M\beta\mu && \text{(4.7a)} \\ \beta y &= C\beta x + N\beta\mu && \text{(4.7b)} \\ S_i(\beta x_i(0), \beta\mu_i) &= \beta x^T(0) P \beta x(0) + \int_0^T \beta\mu_i^T \beta\mu_i \, dt < \beta^2. && \text{(4.7c)} \end{aligned}$$

Now, we let $\widetilde{x} = \beta x$, $\widetilde{y} = \beta y$ and $\widetilde{\mu} = \beta\mu$ and equations (4.7) become

$$\begin{aligned} \dot{\widetilde{x}} &= A\widetilde{x} + B(\beta v) + M\widetilde{\mu} && \text{(4.8a)} \\ \widetilde{y} &= C\widetilde{x} + N\widetilde{\mu} && \text{(4.8b)} \\ S_i(\widetilde{x}_i(0), \widetilde{\mu}_i) &= \widetilde{x}^T(0) P \widetilde{x}(0) + \int_0^T \widetilde{\mu}_i^T \widetilde{\mu}_i \, dt < \beta^2. && \text{(4.8c)} \end{aligned}$$

Equations (4.8) are the new, scaled equations. However, (4.8) are actually the same as (1.12) and (1.13), with a different γ. The theorem presented in Section 4.2.2 finds a bound on this β, which will guarantee separation of the nonlinear output sets using βv.

4.2.1 Review of ODE Theory

In order to address our first result, we must first review some ODE theory. The following theory relies heavily on [28]. The notation has been slightly altered to match the notation used in this thesis.

We start with the ODE

$$\dot{x} = f(x,t),\ x(0) = x_0, \tag{4.9}$$

where $f(x,t)$ is defined on $\mathcal{I} \times \mathcal{X}$. $\mathcal{I}$ is an interval and $\mathcal{X}$ is an open set containing $x(0) = x_0$. Also, $B_\epsilon(x_0)$ is a ball of radius ϵ around x_0. We assume that

1. For each x, f is measurable in t.
2. For each t, f is continuous in x.
3. f is locally Lipschitz in x. That is, for each $x_0 \in \mathcal{X}$, there is an ϵ and nonnegative locally integrable α such that
   (a) $B_\epsilon(x_0) \subset \mathcal{X}$
   (b) $\|f(t,x) - f(t,y)\| \leq \alpha(t)\|x - y\|$ for all $t \in \mathcal{I}$ $x, y \in B_\epsilon(x_0)$.
4. f is locally integrable on t. That is, for each x_0, there is a locally integrable Ω such that $\|f(t,x_0)\| \leq \Omega(t)$

Given the assumptions above, $f(x,t)$ is locally integrable if its norm (absolute value) has a finite integral on each finite subinterval.

Theorem 4.1 *Suppose that ψ is a solution of (4.9) on the interval $[0,\tau] \subset \mathcal{I}$. Then there are numbers $\Delta, c > 0$ so that for every $0 < \delta \leq \Delta$ the following holds: Suppose that $h(x,t)$ satisfies the above assumptions and in addition*

$$\left\| \int_0^t h(\psi(s),s)\, ds \right\| \leq \delta \tag{4.10}$$

for all $t \in [0,\tau]$ and for all η_0 such that $\|\eta_0 - x_0\| \leq \delta$, then

1. *the solution η of $\dot{\eta} = f(t,\eta) + h(t,\eta), \eta(t_0) = \eta_0$ is defined on all of $[0,\tau]$.*
2. *$\|\psi - \eta\|_\infty \leq c\delta$.*

Theorem 4.1 is important because it says, if h satisfies (4.10), then the difference between the solutions to $\dot{x} = f$ and $\dot{x} = f + h$ only differ by a constant times the bound on the difference between the initial conditions of the two systems. Thus if we start with sufficiently close initial conditions, then the solutions will remain close.

4.2.2 Linearizations and Scaled Noise Bounds

We begin this section by presenting our first result, then, instead of going directly to the proof, we will give a brief overview of the idea behind it. The proof is fairly complicated and having the background information will make it more understandable.

Theorem 4.2 *Suppose that v is strictly proper for the linearized system. Using the assumptions from Theorem 4.1, if $\beta \leq \frac{\rho}{2c_3 T \xi c_1^2 \epsilon^2}$, then βv separates the nonlinear output sets, with a noise bound scaled by β^2. Here ρ is the distance between the linearized output sets, c_1 and c_3 are the bounds on the unforced nonlinear equation, and the difference between the forced linear and forced nonlinear equations, with noise, respectively. ξ is the bound on the error term in the linearization and T is the final time.*

The following proof consists of four steps. The first two steps show that, as we scale by β, the solution to the unforced nonlinear problem and the difference between the solutions to the unforced and forced nonlinear problems are linearly bounded in the initial condition and the auxiliary signal and noise, respectively. The third step shows that the difference between the solutions to the forced nonlinear problem and the forced linear problem scales quadratically. The final step finds the bound on β for which βv will guarantee separation of the nonlinear output sets. Figure 4.1 shows the effect of βv on the scaled, nonlinear output sets and how the scaling affects these output sets.

Proof: We start with the nonlinear, unforced system

$$\dot{x} = g(x), \tag{4.11}$$

with initial condition $x_0 = x(0) = g(0) = 0$. Let $B_\epsilon(x_0)$ be a ball of radius ϵ around x_0. Note that $x_0 = 0$ and thus, B is a ball around 0. Now, setting it up as in Theorem 4.1, we let $f(x) = g'(0)x$ (the linearized system) and $h(x) = g(x) - g'(0)x$. If we let ψ be a solution to (4.11) with initial condition $x_0 = 0$, then $\psi = 0$. Note

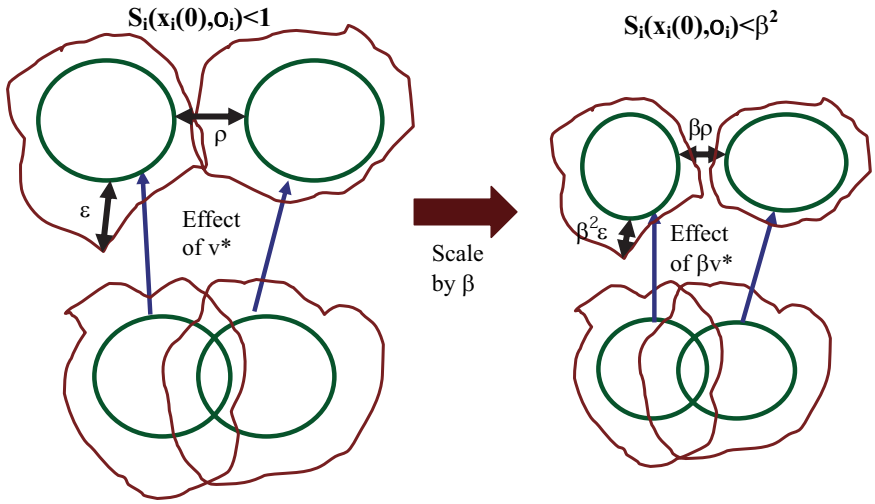


Figure 4.1: Pictorial of Proof.

that $f(x) = g'(0)x$ is continuous and $\|f(0)\| = \|0\| \leq \beta(t)$, for any $\beta(t)$. Thus, we must only show that f is locally Lipschitz in order to have it locally integrable. But, f is linear so

$$\begin{aligned} \|f(x) - f(y)\| &= \|g'(0)x - g'(0)y\| && (4.12) \\ &\leq \|g'(0)\| \, \|x - y\| && (4.13) \end{aligned}$$

and we can let $\alpha(t) = \|g'(0)\|$. In fact $f(x)$ is Lipschitz.

Next, we must show that $\| \int_0^t h(\psi) ds\| \leq \delta$ in order to apply Theorem 4.1. But

$$\begin{aligned} \| \int_0^t h(\psi) \, ds\| &= \| \int_0^t h(0) \, ds\| && (4.14) \\ &= \| \int_0^t (g(0) - g'(0)0) \, ds\| && (4.15) \\ &= \| \int_0^t 0 \, ds\| && (4.16) \\ &= 0 && (4.17) \\ &\leq \delta, && (4.18) \end{aligned}$$

for any $\delta \geq 0$.

Thus f and h satisfy the assumptions in Theorem 4.1. Let η be the solution to

$$\dot{\eta} = f(\eta) + h(\eta) \tag{4.19}$$

and let η_0 be such that

$$\|\eta_0 - x_0\| \leq \delta. \tag{4.20}$$

Then, there exists Δ_1 such that $\|\psi - \eta\|_\infty \leq c_1\delta$ and thus, $\|\eta\|_\infty \leq c_1\delta$ since $\psi = 0$. Now, we note that (4.19), after simplification, is really just $\dot{\eta} = g(\eta)$. Therefore, we can let $\eta = x$ and we get

$$\|x\|_\infty \leq c_1\delta. \tag{4.21}$$

Recall that we are looking at $\delta \leq \Delta_1$ and our $\| \int_0^t h(\psi)\, ds\|$ bound works for any δ. Also, c_1 is independent of δ. We can, therefore, take $\delta = \|\eta_0 - x_0\| = \|x_0\|$, since $\eta_0 = 0$. Hence,

$$\|x\|_\infty \leq c_1\|x_0\| \tag{4.22}$$

and we have shown that $\|x\|_\infty$ is linearly bounded in $\|x_0\|$. Note, that since we are looking at $x_0 \in B_\epsilon(0)$, then $\|x_0\| \leq \epsilon$. Therefore, (4.22) becomes

$$\|x\|_\infty \leq c_1\epsilon. \tag{4.23}$$

This case does dissolve into a degenerate case since we are basically saying that $0 \leq 0$, but Theorem 4.1 tells us that as long as we have a bounded solution (i.e. $\psi = 0$), then all solutions starting with a close initial condition (i.e. inside a ball of radius ϵ) will also stay bounded.

Next, we will show that $\|x - \widetilde{x}\|$ is linearly bounded in $\|v\|$ and $\|\mu\|$, where x is the solution to (4.11) and $\widetilde{x}$ is the solution to

$$\dot{\widetilde{x}} = g(\widetilde{x}) + Bv + M\mu,\ x(0) = \widetilde{x}(0) \tag{4.24}$$

We let $f(x) = g(\widetilde{x})$ and $h(x) = Bv + M\mu$. First, note that h has no dependence on x, so $h(x) = h = Bv + M\mu$. Again, $f(x) = g(\widetilde{x})$ is continuous and $\|f(x_0)\| = \|g(0)\| = 0 \leq \beta(t)$. From our assumptions, $f = g(\widetilde{x})$ is locally Lipschitz, so all we need to show

is that $\|h(\psi)\ ds\| \leq \delta$, where ψ is a solution of $\dot{\widetilde{x}} = g(\widetilde{x})$. But

$$\begin{aligned} \| \int_0^t h(\psi) ds\| &= \| \int_0^t (Bv + M\mu) ds\| && (4.25) \\ &\leq \int_0^t |Bv| ds + \int_0^t |M\mu| ds && (4.26) \\ &\leq \left(\int_0^t \|B\|^2 ds \right)^{\frac{1}{2}} \left(\int_0^t \|v\|^2 ds \right)^{\frac{1}{2}} + \left(\int_0^t \|M\|^2 ds \right)^{\frac{1}{2}} \left(\int_0^t \|\mu\|^2 ds \right)^{\frac{1}{2}} \\ &= \|B\|_2 \sqrt{T} \|v\|_2 + \|M\|_2 \sqrt{T} \|\mu\|_2. && (4.27) \end{aligned}$$

From our assumptions in Theorem 4.2, we know that $\|v\|_2 \leq \frac{\delta}{2\|B\|_2\sqrt{T}}$ and $\|\mu\|_2 \leq \frac{\delta}{2\|M\|_2\sqrt{T}}$, thus each term in (4.27) is less than or equal to $\frac{\delta}{2}$. Therefore,

$$\|B\|_2 \sqrt{T} \|v\|_2 + \|M\|_2 \sqrt{T} \|\mu\|_2 \leq \delta. \tag{4.28}$$

Now, following Theorem 4.1, for all η_0 such that $\|\eta_0 - \widetilde{x}_0\| \leq \delta$, there exists Δ_2 such that

$$\|\psi - \eta\|_\infty \leq c_2 \delta, \tag{4.29}$$

where η is the solution to $\dot{\eta} = f(\eta) + h(\eta) = g(\eta) + Bv + M\mu$. Recall that ψ is the solution to $f(\psi) = g(\psi)$. So, we can let $\psi = x$ and $\eta = \widetilde{x}$. Then (4.29) becomes

$$\|x - \widetilde{x}\|_\infty \leq c_2 \delta. \tag{4.30}$$

Bound (4.30) works for all $\delta \leq \Delta_2$, in particular, we take $\delta = \|B\|_2\sqrt{T}\|v\|_2 + \|M\|_2\sqrt{T}\|\mu\|_2$ from (4.28). Therefore, we have

$$\|x - \widetilde{x}\|_\infty \leq c_2(\|B\|_2 \sqrt{T} \|v\|_2 + \|M\|_2 \sqrt{T} \|\mu\|_2) \tag{4.31}$$

and thus, $\|x - \widetilde{x}\|_\infty$ is linearly bounded in both $\|v\|$ and $\|\mu\|$. Because $\|x\|_\infty$ is linearly bounded in $\|x_0\|$ and $\|x - \widetilde{x}\|_\infty$ is linearly bounded in $\|v\|$ and $\|\mu\|$, we can conclude that $\|\widetilde{x}\|_\infty$ is linearly bounded in $\|\widetilde{x}_0\|$ as well.

The third step is to show that $\|\widehat{x} - \widetilde{x}\|_\infty$ is quadratically bounded in $\|\widetilde{x}\|_\infty$, where $\widehat{x}$ is the solution to

$$\dot{\widehat{x}} = A\widehat{x} + Bv + M\mu \tag{4.32}$$

and $\widetilde{x}$ is the solution to

$$\dot{\widetilde{x}} = A\widetilde{x} + Bv + M\mu + O(\|\widetilde{x}\|^2). \tag{4.33}$$

First, note that $\widetilde{x}$ from (4.33) is the same as $\widetilde{x}$ from (4.24), we have just rewritten (4.24) using Taylor's Theorem and linearizing around $\overline{x} = 0$. We use the initial condition $\widehat{x}(0) = \widetilde{x}(0)$. Let $f(x) = A\widetilde{x} + Bv + M\mu$ and $h(x) = O(\|\widetilde{x}\|^2)$. Clearly, since f is a linear transformation, we know that f is continuous. In order to show that $f(x_0) = Ax_0 + Bv + M\mu$ is locally integrable, we note that we are measuring v and μ in L^2 and we are on a finite interval. Thus, $f(x_0) \in L^1$, which implies that f is locally integrable. We must also show that f is locally Lipschitz.

$$\begin{aligned}
\|f(x) - f(y)\| &= \|Ax + Bv + M\mu - (Ay + Bv + M\mu)\| && (4.34)\\
&= \|Ax - Ay\| && (4.35)\\
&= \|A(x - y)\| && (4.36)\\
&\leq \|A\| \, \|x - y\|. && (4.37)
\end{aligned}$$

If we let $\alpha(t) = \alpha = \|A\|$, f is Lipschitz.

Now, we must show that $h(x) = O(\|\widetilde{x}\|^2)$ satisfies the bound from Theorem 4.1. Note that by definition, $O(\|\widetilde{x}\|^2) \leq \xi\|\widetilde{x}\|^2$ for some constant ξ and for small enough $\widetilde{x}$. This means there exists Δ_3 so that for $O(\|\widetilde{x}\|^2) \leq \Delta_3$, we have that $O(\|\widetilde{x}\|^2) \leq \xi\|\widetilde{x}\|^2$. Then

$$\begin{aligned}
\| \int_0^t h(\psi) ds\| &= \| \int_0^t O(\|\widetilde{x}\|^2) ds\| && (4.38)\\
&\leq \| \int_0^t \xi\|\widetilde{x}\|^2 ds\| && (4.39)\\
&= \xi \int_0^t \|\widetilde{x}\|^2 ds && (4.40)\\
&\leq \xi \int_0^T \|\widetilde{x}\|^2 ds && (4.41)\\
&= \xi\|\widetilde{x}\|_2^2 && (4.42)\\
&\leq \xi T\|\widetilde{x}\|_\infty^2. && (4.43)
\end{aligned}$$

Recall from the previous step that $\|\widetilde{x}\|_\infty$ is linearly bounded in $\|\widetilde{x}_0\|$. Thus, h satisfies the bound (4.10) and we can apply Theorem 4.1.

We have that η is the solution to $\dot{\eta} = f(\eta) + h(\eta) = A\eta + Bv + M\mu + O(\|\eta\|^2)$ and ψ is a solution to $f(\psi) = A\psi + Bv + M\mu$. Applying Theorem 4.1, there exists Δ_3 (note it's the same Δ_3 as above) such that $\|\eta - \widetilde{x}_0\| \leq \delta$, and thus

$$\|\psi - \eta\|_\infty \leq c_3\delta. \tag{4.44}$$

We let $\psi = \widehat{x}$ and $\eta = \widetilde{x}$ and we get that (4.44) is

$$\|\widehat{x} - \widetilde{x}\|_\infty \leq c_3\delta. \tag{4.45}$$

Since h satisfies the bound from Theorem 4.1, we can use $\delta = T\xi\|\widetilde{x}\|_\infty^2$ and (4.45) becomes

$$\|\widehat{x} - \widetilde{x}\|_\infty \leq c_3 T\xi\|\widetilde{x}\|_\infty^2, \tag{4.46}$$

and thus, $\|\widehat{x} - \widetilde{x}\|_\infty$ is quadratic in $\|\widetilde{x}\|$.

We have now shown that as the problem is shrunk by a parameter β, the linear output sets converge to the zero sets linearly in β while the nonlinear output sets converge to the linear output sets quadratically, using $\Delta = \min_{i=1,2,3}\{\Delta_i\}$. This means that before the linear output sets become the zero set, the nonlinear output sets will be separated by the application of βv. The final step is to compute a bound for β.

Let v^* be a strictly proper input signal for the linear equation

$$\dot{x}_i = Ax_i + Bv + M\mu_i, \;\; x_i(0) = x_{0i} \tag{4.47}$$

with noise bound

$$S(x_{i,0}, \mu_i) = x_{0i}^T Q x_{0i} + \int_0^T \|\mu_i(t)\|^2 \, dt < 1. \tag{4.48}$$

Since v^* is strictly proper, there exists a positive distance ρ between the two output sets.

Now suppose we scale the noise bound by β^2 (note that this bound is a squared norm). From the first two steps of the proof, we know that the linear equations are linearly bounded in $\|x_{0,i}\|, \|\mu_i\|_2$ and $\|v\|_2$. Thus, βv^* is proper for the scaled, linear equation and the distance between the faulty and working output sets is $\beta\rho$. From the previous bound, we know that the distance between the linear and nonlinear

output sets scales quadratically in β. Thus, after scaling, this distance is less than or equal to $\beta^2 c_3 T\xi \|\widetilde{x}\|_\infty^2$. Now, since $\|\widetilde{x}\|_\infty$ is linearly bounded in $\|\widetilde{x}_0\|$, we know that $\|\widetilde{x}\|_\infty^2 \leq c_1^2 \|\widetilde{x}_0\|^2$. But, $\widetilde{x}_0 \in B_\epsilon(0)$, thus $\|\widetilde{x}_0\|^2 \leq \epsilon^2$. Hence, the distance between the scaled nonlinear and linear output sets is less than or equal to $\beta^2 c_3 T\xi c_1^2 \epsilon^2$.

We would like to find a β which guarantees separation of the nonlinear output sets. Therefore, we need to find β such that

$$\frac{\beta\rho}{2} \leq \beta^2 c_3 T\xi c_1^2 \epsilon^2. \tag{4.49}$$

Equation (4.49) says that half the distance between the linear outputs sets is greater than the maximum distance between the linear and nonlinear output sets. Solving (4.49) for β, we get that

$$\beta \leq \frac{\rho}{2c_3 T\xi c_1^2 \epsilon^2}. \tag{4.50}$$

□

4.3 Conditions for Which linear v is Proper for Nonlinear Problem

In this section, we address the important question of when a proper v from the linear system can be used on the nonlinear system. Unlike Section 4.2.2, where we parameterized the entire problem to find a proper auxiliary signal which would guarantee detection in the nonlinear problem, for this question, we find a proper v for the linear problem and then derive conditions for which this v is proper for the nonlinear problem, except for possibly tighter noise bounds. Case studies suggest that this is a reasonable question to ask [26, 1]. In [26] it is shown that linearizations often work well, even for highly nonlinear systems and also that in certain cases, using the optimal v from the linear system, one could allow more noise in the nonlinear system then the linearized system. Linearizations also work well for incipient faults [1]. In this section we will assume that the noises in the ODE and the output equation are independent (i.e. The same μ_i is not present in both the ODE and the output

equation). Since this assumption allows for more noise, a test signal that is proper under this assumption will be proper if μ_0, μ_1 are not independent.

It suffices to first consider a single nonlinear system which may be the faulty or nonfaulty system:

$$\dot{x} = f(x) + Bv + M\mu_1 \tag{4.51a}$$
$$y = g(x) + N\mu_2, \tag{4.51b}$$

and note that, after linearization around zero, this system can be written as

$$\dot{x} = Ax + E_1(x) + Bv + M\mu_1 \tag{4.52a}$$
$$y = Cx + E_2(x) + Dv + N\mu_2 \tag{4.52b}$$

where $E_i(0) = 0$ and $E_i(x)$ are the linearization errors.

The linear system gotten by neglecting the $E_i(x)$ is

$$\dot{x} = Ax + Bv + M\phi_1 \tag{4.53a}$$
$$y = Cx + Dv + N\phi_2. \tag{4.53b}$$

We will use a noise bound of $S(x(0), \phi) \leq \delta$. Using the noise bounds, we find a proper v^* for our two linear systems. The goal is then to find noise bounds for the two nonlinear systems (4.52), for which v^* is proper.

Now we will assume that M, N are invertible and we rewrite (4.52) as

$$\dot{x} = Ax + Bv + M(M^{-1}E_1(x) + \mu_1) \tag{4.54a}$$
$$y = Cx + Dv + N(N^{-1}E_2(x) + \mu_2). \tag{4.54b}$$

We can treat the nonlinearities E_i as extra noise in our system and equate the noises from (4.53) to the noises in (4.54). Thus, we have $\phi_1 = M^{-1}E_1(x) + \mu_1$ and $\phi_2 = N^{-1}E_2(x) + \mu_2$.

Going back to our noise bound, we know that $S(x(0), \phi) \leq \delta$, where $S(x(0), \phi) = \|x(0)\|^2 + \int_0^T \phi^T \phi \, dt \leq \delta$. Since we have split ϕ into ϕ_1 and ϕ_2, we can rewrite our

noise bound as

$$\begin{aligned} \|x(0)\|_Q^2 + \int_0^T \begin{bmatrix} \phi_1^T & \phi_2^T \end{bmatrix} \begin{bmatrix} \phi_1^T \\ \phi_2^T \end{bmatrix} dt &= \|x(0)\|_Q^2 + \int_0^T (\phi_1^T \phi_1 + \phi_2^T \phi_2) dt \\ &= \|x(0)\|_Q^2 + \int_0^T (\|\phi_1\|^2 + \|\phi_2\|^2) dt \quad (4.55) \\ &< \delta, \quad (4.56) \end{aligned}$$

where $\| * \|_Q^2 = *^T Q *$.

Substituting $\phi_1 = M^{-1}E_1(x) + \mu_1$ and $\phi_2 = N^{-1}E_2(x) + \mu_2$ into this inequality, we have

$$\|x(0)\|_Q^2 + \int_0^T (\|M^{-1}E_1(x) + \mu_1\|^2 + \|N^{-1}E_2(x) + \mu_2\|^2) dt < \delta. \quad (4.57)$$

Equation (4.57) measures the distance between $(x(0), \mu_1, \mu_2)$ and $(0, M^{-1}E_1, N^{-1}E_2)$ in an inner product norm. Thus, if

$$\begin{aligned} \|M^{-1}E_1(x), N^{-1}E_2(x)\|_2^2 &= \int_0^T \|M^{-1}E_1(x) + \mu_1\|^2 + \|N^{-1}E_2(x) + \mu_2\|^2 dt \\ &< p < \delta \quad (4.58) \end{aligned}$$

and

$$\|x(0)\|_Q^2 + \int_0^T \|\mu_1\|^2 + \|\mu_2\|^2 dt < (\sqrt{\delta} - \sqrt{p})^2 \quad (4.59)$$

then (4.56) holds. Thus, if the nonlinearities for the normal and faulty systems satisfy (4.58) with p_0, p_1, then the v computed from the linearizations with noise bounds (4.56) with δ_0, δ_1 will be proper for the nonlinear systems with noise bounds $\sqrt{\delta_i} - \sqrt{p_i}$.

We note that we require M, N to be invertible, however, we can be less restrictive under certain circumstances.

Corollary 4.3 *Assuming $E_1(x)$ and $E_2(x)$ are in the ranges of M and N, respectively, then M and N need not necessarily be invertible, there must only exist the Moore-Penrose psuedoinverse, $M^\dagger$ and $N^\dagger$.*

Proof: We will only present the proof for $E_1(x)$ and M. The same process is used for $E_2(x)$ and N. We know that $MM^\dagger$ is the orthogonal projection onto the range of M, but $E_1(x)$ is in the range of M [29]. Thus, $M(M^\dagger E_1(x)) = MM^\dagger E_1(x) = E_1(x)$. □

4.4 Computational Study: The Oscillating Pendulum

To illustrate the previous results we consider a pendulum. The nonlinearities are typical of those arising in a number of mechanical systems including some in robotics. An oscillating pendulum is governed by the principle of angular momentum, which states that the time rate of change of angular momentum about any point is equal to the moment of the resultant force about that point [30]. Figure 4.2 gives the typical set up for a pendulum. L is the length of the weightless rod, attached at the origin O. At the bottom of the rod is a ball of mass m. Let θ be the angle between the rod and the downward vertical direction and we will take the counterclockwise direction to be positive. The damping force $c\dot{\theta}$ occurs in the opposite direction of motion. The angular momentum about the origin is defined to be $mL^2\dot{\theta}$. The moment of resultant force is the sum of the moment arms of resistive force and gravitational force, $cL\dot{\theta}$ and $mgL\sin\theta$ respectively. Thus, the equation of motion of the pendulum is

$$mL^2\ddot{\theta} = -(cL\dot{\theta} + mgL\sin\theta). \tag{4.60}$$

Note that the right hand side of (4.60) is negative because the forces make the pendulum move in a clockwise position and we take counterclockwise as the positive direction. Rearranging, we have

$$\ddot{\theta} + \frac{c}{mL}\dot{\theta} + \frac{g}{L}\sin\theta = 0. \tag{4.61}$$

Now, since we would like add an auxiliary input signal, we replace the right hand side of (4.61) with bu, where u is the torque at the origin (i.e. the joint where the pendulum is attached). Thus, the final nonlinear equation is

$$\ddot{\theta} + \frac{c}{mL}\dot{\theta} + \frac{g}{L}\sin\theta = bu. \tag{4.62}$$

The next step is to take the second order ODE and replace it with a system of first order ODE's. To do this, we let $x_1 = \theta$ and $x_2 = \dot{\theta}$. Taking the derivative of

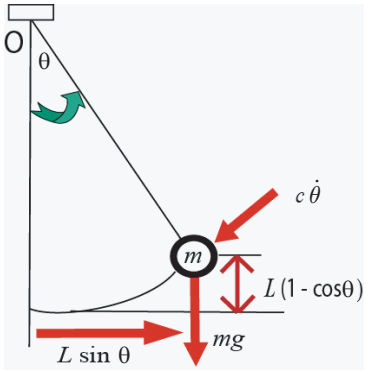


Figure 4.2: Pendulum Setup.

each, we have $\dot{x}_1 = \dot{\theta}$ and $\dot{x}_2 = \ddot{\theta}$. Our final system of equations is

$$\begin{aligned} \dot{x}_1 &= x_2 && \text{(4.63a)} \\ \dot{x}_2 &= bu - \frac{\mathrm{g}}{L}\sin x_1 - \frac{c}{mL}x_2 + M\mu && \text{(4.63b)} \\ y &= Cx + N\mu && \text{(4.63c)} \end{aligned}$$

The parameters given in [30] are $\frac{\mathrm{g}}{L} = 9$ and $\frac{c}{mL} = \frac{1}{5}$. We also note that $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$ since $x_1 = \theta$ is angle between the rod and the downward vertical motion, where as $x_2 = \dot{\theta}$ is the rate of change of this angle. Using these facts, we have

$$\begin{aligned} \dot{x}_1 &= x_2 && \text{(4.64a)} \\ \dot{x}_2 &= bu - 9\sin x_1 - \frac{1}{5}x_2 + M\mu, && \text{(4.64b)} \\ y &= x_1 + N\mu && \text{(4.64c)} \end{aligned}$$

and we note that (4.64) is in the form of (4.1).

4.4.1 Scaled Linearization

Now we have the system set up in the appropriate form so we can begin calculating the β bound. Using the notation from Section 4.2.2, $g(x) = \begin{bmatrix} x_2 \\ -9\sin x_1 - \frac{1}{5}x_2 \end{bmatrix}$ and we note that $g(x)$ satisfies the assumption that $g(0) = \begin{bmatrix} 0 \\ -9\sin 0 - 0 \end{bmatrix} = 0$. We will

assume that we are inside a ball of radius 1, i.e. $\epsilon = 1$ and we have that $x_0 = 0$. For the first step of the proof, we start with the unforced system $\dot{x} = g(x)$ and we let $f(x) = g'(0)x$ (the linearization of $g(x)$ around 0).

$$g'(0) = \begin{bmatrix} 0 & 1 \\ -9\cos 0 & -\frac{1}{5} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -9 & -\frac{1}{5} \end{bmatrix}$$

and thus

$$f(x) = g'(0)x = \begin{bmatrix} x_2 \\ -9x_1 - \frac{1}{5}x_2 \end{bmatrix}. \tag{4.65}$$

Now, we let $h(x) = g(x) - g'(0)x$. Then

$$\begin{aligned} g(x) - g'(0)x &= \begin{bmatrix} x_2 \\ -9\sin x_1 - \frac{1}{5}x_2 \end{bmatrix} - \begin{bmatrix} x_2 \\ -9x_1 - \frac{1}{5}x_2 \end{bmatrix} && (4.66) \\ &= \begin{bmatrix} 0 \\ 9(x_1 - \sin x_1) \end{bmatrix}. && (4.67) \end{aligned}$$

From the first step of our proof of Theorem 4.2, we know that $\|x\|_\infty \leq c_1\epsilon$. We have that $\epsilon = 1$ and to compute c_1, we need to find a bound on $\dot{x} = g(x)$. In order to do this, we will find a Lyapunov function to bound the solution. If we take $V(x_1, x_2) = x_2^2 + 18(1 - \cos x_1)$ and show that $V' \leq 0$, then it will be an acceptable function.

$$\begin{aligned} V'(x_1, x_2) &= 2x_2\dot{x}_2 + 18\sin x_1\dot{x}_1 && (4.68) \\ &= 2x_2(-9\sin x_1 - \frac{1}{5}x_2) + 18x_2\sin x_1 && (4.69) \\ &= -18x_2\sin x_1 - \frac{2}{5}x_2^2 + 18x_2\sin x_1 && (4.70) \\ &= -\frac{2}{5}x_2^2. && (4.71) \end{aligned}$$

We see from (4.71) that $V' \leq 0$ for all $x_1, x_2 \in B_1(x_0)$. Now, we note that V can be written as $V(x_1, x_2) = x_2^2 + 9x_1^2 + O(\|x_1\|^4)$ and therefore V is ellipse like. This means that we can bound the solution to $\dot{x} = g(x)$ inside this ellipse. Using Maple, we find that the cross section of V with the ball of radius 1 occurs at $x_2 = \pm\sqrt{c_1}$. We see from figure 4.3 that $\sqrt{c_1}$ is the long side of the ellipse and it touches $B_1(x_0)$

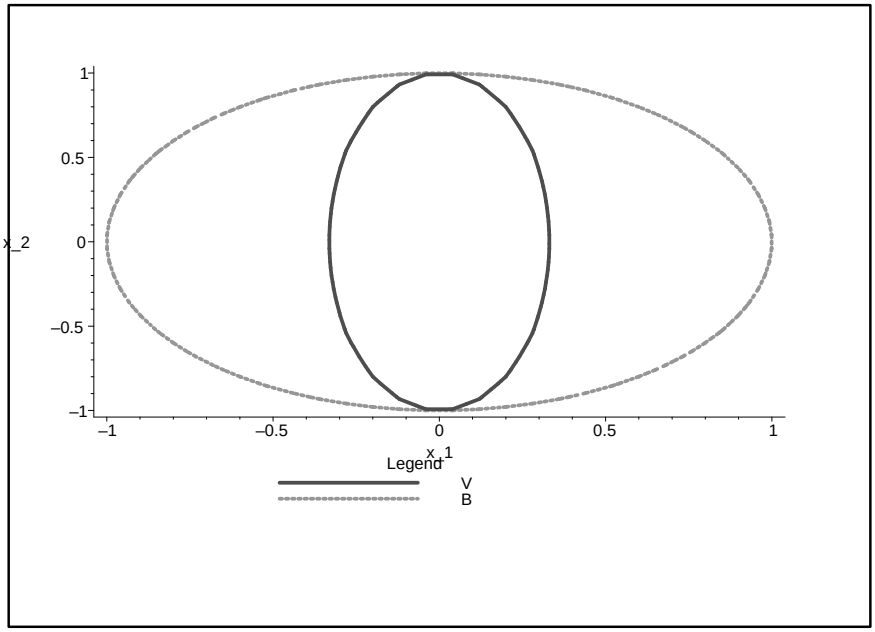


Figure 4.3: Cross Section of V with $B_1(x_0)$.

at $x_2 = 1$, which means we can take $c_1 = 1$. Figure 4.4 shows the idea behind the bound. If we call the inner circle c_{new}, the ellipse c_1 and the outer circle $c_0 = 1$, then what we are saying is that if we start with the bound $\|x_0\| \leq c_{new}$, then $V \leq c_1$ and $\|x(t)\| \leq c_0 = 1$, and thus our solution will be bounded by 1. Doing it this way, we tighten our bound in the initial condition to be $\epsilon_{new} = c_{new}$. Now, we need to find c_{new}, which is just the x_1 coordinate when $x_2 = 0$, with $c_1 = 1$. So, dropping the error term in V, we have $V(x_1, 0) = 1 = 9x_1^2$ and thus, $x_1 = \pm\frac{1}{3}$. Finally we have that $x_1 = c_{new} = \epsilon_{new} = \frac{1}{3}$.

Then next step is to find c_3, which is the bound on the difference between the forced linearized equation with noise and the forced nonlinear equation with noise, i.e. the difference between

$$\dot{\widetilde{x}} = \begin{bmatrix} 0 & 1 \\ -9 & -\frac{1}{5} \end{bmatrix} \widetilde{x} + Bv + M\mu \tag{4.72}$$

and

$$\dot{\widehat{x}} = \begin{bmatrix} 0 & 1 \\ -9 & -\frac{1}{5} \end{bmatrix} \widehat{x} + Bv + M\mu + O(\|\widehat{x}\|^2), \tag{4.73}$$

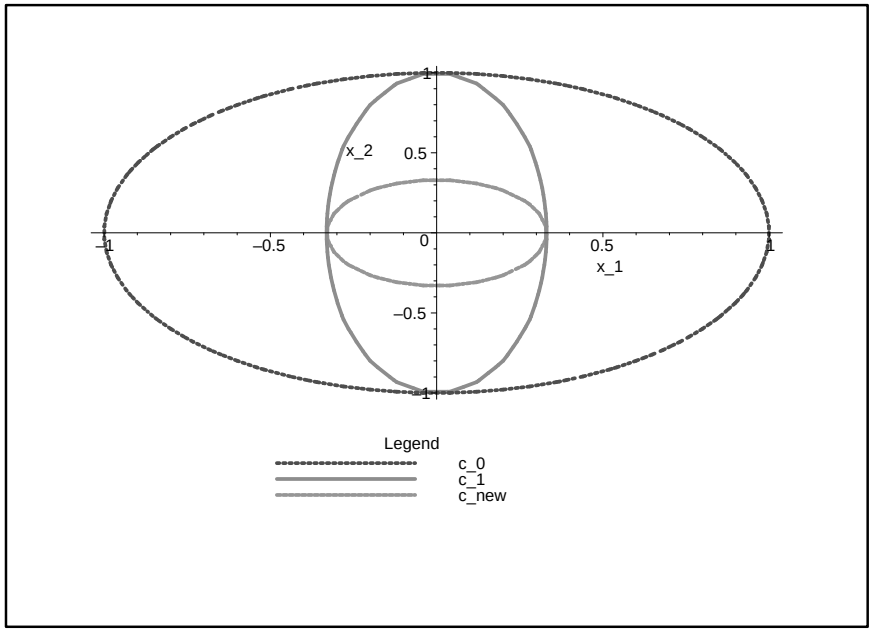


Figure 4.4: Concentric circles.

with initial condition $\widetilde{x}_0 = \widehat{x}_0$. If we let $\omega = \widetilde{x} - \widehat{x}$, then the difference between (4.72) and (4.73) is simply the equation

$$\dot{\omega} = \begin{bmatrix} 0 & 1 \\ -9 & -\frac{1}{5} \end{bmatrix} \omega + O(\|\widehat{x}_1\|^2) \tag{4.74}$$

with initial condition $\omega_0 = 0$. The bound on the solution to (4.74), ω, is our c_3.

Note that (4.74) is a linear ODE with a force applied to it. Therefore, the solution is

$$\omega = e^{At}\omega_0 + \int_0^t e^{A(t-s)} O(\|\widehat{x}\|^2)\, ds \tag{4.75}$$

$$= \int_0^t e^{A(t-s)} O(\|\widehat{x}\|^2)\, ds, \tag{4.76}$$

since $\omega_0 = 0$. Thus,

$$\|\omega\| = \| \int_0^t e^{A(t-s)} O(\|\widehat{x}\|^2)\, ds \| \tag{4.77}$$

$$\leq \int_0^t \|e^{A(t-s)}\|\, \|O(\|\widehat{x}\|^2)\|\, ds. \tag{4.78}$$

We bound the error term in (4.78) by noting that this term comes from the linearization of $\sin x$, and so $O(\|\widehat{x}\|^2) \leq \frac{\widetilde{x}^3}{6} + \ldots$. Figure 4.5 shows the plot of $\sin x - x$ on $x = -1$ to $x = 1$, and we see that $\|\sin x - x\| \leq 0.16$ for $\|x\| \leq 1$. Therefore, $O(\|\widetilde{x}\|^2) \leq 0.16$ for $\|x\| \leq 1$.

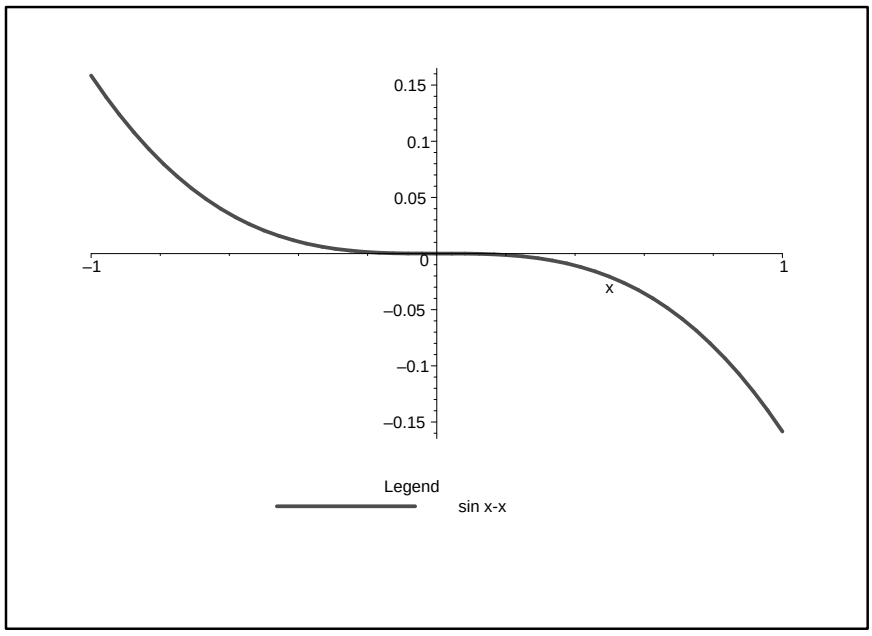


Figure 4.5: Error of Linearization.

Returning to (4.78), we have that

$$\|\omega\| \leq 0.16 \int_0^t \|e^{A(t-s)}\| ds. \tag{4.79}$$

The eigenvalues of A are $-0.1 \pm 2.9983i$, so A is a stable matrix. Thus, we can write

A as $A = LPL^{-1}$, where

$$L = \begin{bmatrix} -0.0105 - 0.3161i & -0.0105 + 0.3161i \\ 0.9487 & 0.9487 \end{bmatrix}$$

is the matrix of eigenvectors and

$$P = \begin{bmatrix} -0.1 + 2.9982i & 0 \\ 0 & -0.1 - 2.9983i \end{bmatrix}$$

is the diagonal matrix of eigenvectors and $e^{A(t-s)}$ can be written as

$$e^{A(t-s)} = L \begin{bmatrix} e^{(-0.1+2.9982i)(t-s)} & 0 \\ 0 & e^{(-0.1-2.9983i)(t-s)} \end{bmatrix} L^{-1}$$

. Therefore,

$$\begin{aligned} \|e^{A(t-s)}\| &= \left\| L \begin{bmatrix} e^{(-0.1+2.9982i)(t-s)} & 0 \\ 0 & e^{(-0.1-2.9983i)(t-s)} \end{bmatrix} L^{-1} \right\| && (4.80) \\ &\leq \|L\| \left\| \begin{bmatrix} e^{(-0.1+2.9982i)(t-s)} & 0 \\ 0 & e^{(-0.1-2.9983i)(t-s)} \end{bmatrix} \right\| \|L^{-1}\| && (4.81) \\ &= 3.0021 \left\| \begin{bmatrix} e^{(-0.1+2.9982i)(t-s)} & 0 \\ 0 & e^{(-0.1-2.9983i)(t-s)} \end{bmatrix} \right\| && (4.82) \end{aligned}$$

Now, we note that the matrix in (4.82) is a diagonal matrix with each entry of the form $e^{(a\pm bi)(t-s)}$. Also, we know that $\|e^{(a\pm bi)(t-s)}\| = \|e^{a(t-s)}\|$ since $\|e^{\pm bi}\| = 1$. But, in this case, $a < 0$ and thus, $\|e^{a(t-s)}\| \leq 1$ since $s \leq t$. Therefore, (4.82) is less than or equal to 3.0021*1=3.0021.

Returning to (4.79), we have

$$\begin{aligned} \|\omega\| &\leq 0.16 \int_0^t 3.0021 ds && (4.83) \\ &= 0.4803 s|_{s=0}^{s=t} && (4.84) \\ &= 0.4803 t. && (4.85) \end{aligned}$$

We are on the interval $[0, 5]$, so (4.85) is less than or equal to 2.4015, and thus $c_3 = 2.4015$.

The final term we need in order to compute the β bound is ξ, which is just the bound on $O(\|\widetilde{x}\|^2)$ from the linearization of $\sin x_1$. Recall that in the previous step, we said that $O(\|\widetilde{x}\|^2) \leq 0.16$, for $\|x\| \leq 1$. Thus, $\xi = 0.16$.

Therefore, putting it all together our β bound is

$$\begin{aligned} \beta &\leq \frac{\rho}{2 * 2.4015 * 5 * 0.16 * 1^2 * (\frac{1}{3})^2} && (4.86) \\ &= \frac{\rho}{0.4269}. && (4.87) \end{aligned}$$

(4.87) is the β bound for the properly working model, but we still have to compute the bound for the faulty model. We will take the minimum of these two bounds as the β bound for the problem. We assume there is degradation in the lubricant, thus we increase the friction c. We take $c = 5$ and proceed as before, but with the system

$$\begin{aligned} \dot{x}_1 &= x_2 && (4.88a) \\ \dot{x}_2 &= bu - 9\sin x_1 - x_2 + M\mu, && (4.88b) \\ y &= x_1 + N\mu. && (4.88c) \end{aligned}$$

For simplicity, we use the same Lyapunov function $V(x_1, x_2) = x_2^2 + 18(1 - \cos x_1)$ as above to bound

$$\dot{x} = g(x) = \begin{bmatrix} x_2 \\ -9\sin x_1 - x_2 \end{bmatrix}.$$

Again, we must only show that $V'(x_1, x_2) \leq 0$.

$$\begin{aligned} V'(x_1, x_2) &= 2x_2\dot{x}_2 + 18\sin x_1\dot{x}_1 && (4.89) \\ &= 2x_2(-9\sin x_1 - x_2) + 18x_2\sin x_1 && (4.90) \\ &= -18x_2\sin x_1 - 2x_2^2 + 18x_2\sin x_1 && (4.91) \\ &= -2x_2^2. && (4.92) \end{aligned}$$

Without going through the process again, since $V'(x_1, x_2) \leq 0$ and it is the same one as the proper model, we know that $\epsilon_{new} = \frac{1}{3}$ and $c_1 = 1$.

The next step is to compute c_3. If we let $\omega = \widetilde{x} - \widehat{x}$, then c_3 is the bound on the solution to

$$\dot{\omega} = \begin{bmatrix} 0 & 1 \\ -9 & -1 \end{bmatrix} \omega + O(\|\widehat{x}\|^2), \quad (4.93)$$

with initial condition $\omega_0 = 0$. As before, we know that the bounded solution is

$$\|\omega\| \leq \int_0^t \|e^{A(t-s)}\| \, \|O(\|\widehat{x}\|^2)\| \, ds \tag{4.94}$$

and $\|O(\|\widehat{x}\|^2)\| \leq 0.16$. Therefore,

$$\|\omega\| \leq 0.16 \int_0^t \|e^{A(t-s)}\| \, ds. \tag{4.95}$$

The eigenvalues of A are $-0.5 \pm 2.9580i$ and thus, A is stable. Therefore, $A = LPL^{-1}$, where

$$L = \begin{bmatrix} -0.0527 - 0.3118i & -0.0527 + 0.3118i \\ 0.9487 & 0.9487 \end{bmatrix},$$

$$P = \begin{bmatrix} -0.5 + 2.9580i & 0 \\ 0 & -0.5 - 2.9580i \end{bmatrix},$$

and $\|e^{A(t-s)}\| \leq \|L\|\|e^{P(t-s)}\|\|L^{-1}\| = 3.0531\|e^{P(t-s)}\|$. Using the same rationale as in the properly working case, since $a = -0.5 \leq 0$, we have $\|e^{P(t-s)}\| \leq 1$. Substituting this into (4.95), we have

$$\begin{aligned} \|\omega\| &\leq 0.16 \int_0^t 3.0531 ds & (4.96) \\ &= 0.4885 \int_0^t ds & (4.97) \\ &= 0.4885 s|_{s=0}^{s=t} & (4.98) \\ &= 0.4885 t. & (4.99) \end{aligned}$$

(4.99) is less than 2.4425 since we are on the interval $[0, 5]$. Therefore, $c_3 = 2.4425$ and

$$\begin{aligned} \beta &= \frac{\rho}{2 * 2.4425 * 5 * 0.16 * 1^2 * (\frac{1}{3})^2} & (4.100) \\ &= \frac{\rho}{0.4342}. & (4.101) \end{aligned}$$

We have computed the β bound for both the proper model and the faulty model and we will use the minimum of these for the bound for the problem. Recall that the

proper model bound, from (4.87), is $\beta_{proper} = \frac{\rho}{0.4269}$ and we just computed the faulty model β bound to be $\beta_{faulty} = \frac{\rho}{0.4342}$. Therefore, the final bound for the problem is $\beta = \beta_{faulty} = \frac{\rho}{0.4342}$.

Putting it all together and including the noise, our two systems are

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -9 & -\frac{1}{5} \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 & 0 \\ 0 & 0.5 \end{bmatrix} \mu, \tag{4.102a}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x + \begin{bmatrix} 0.25 & 0 \end{bmatrix} \mu, \tag{4.102b}$$

to model the properly working system and

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -9 & -1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 & 0 \\ 0 & 0.5 \end{bmatrix} \mu, \tag{4.103a}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x + \begin{bmatrix} 0.25 & 0 \end{bmatrix} \mu, \tag{4.103b}$$

to model the faulty system.

Using SOCS (Sparse Optimal Control Software) [13] we find the distance between the linear output sets, using a strictly proper v from the linearization, is $\rho = 0.129$. Thus, $\beta = \frac{0.129}{0.4342} = 0.2971$. We would like to find the noise bound for the nonlinear problem, which is simply $\beta^2 = 0.08827$. Theorem 4.2 guarantees that βv will separate the nonlinear output sets with a noise bound of β^2. Using SOCS we computed the actually noise bound that holds for βv with $\beta = 0.2971$ and find the bound to be 0.12788506. We note that since $\beta^2 \leq 0.12788506$, βv is proper for the nonlinear problem, as predicted by our theorem.

4.4.2 Results for Tighter Noise Bound

For the second result, we are not concerned with either the properly working or faulty model, only the noise bound. So, for simplicity, we write our system (4.64) as

$$\dot{x} = Ax + E_1(x) + Bu + \begin{bmatrix} 0 & 0 \\ 0 & 0.5 \end{bmatrix} \mu_1 \tag{4.104a}$$

$$y = Cx + \begin{bmatrix} 0.25 & 0 \end{bmatrix} \mu_2, \tag{4.104b}$$

where $E_1(x)$ is the error from linearization and we note that $E_2(x) = 0$ since there is no nonlinearity in the output equation. Recall that the noise from (4.104a) is independent of the noise in (4.104b), i.e. $\mu_1 = \begin{bmatrix} 0 \\ \mu_{12} \end{bmatrix}$ and $\mu_2 = \begin{bmatrix} \mu_{21} \\ 0 \end{bmatrix}$. Our linearized system is

$$\dot{x} = Ax + Bu + \begin{bmatrix} 0 & 0 \\ 0 & 0.5 \end{bmatrix} \phi_1 \tag{4.105a}$$

$$y = Cx + \begin{bmatrix} 0.25 & 0 \end{bmatrix} \phi_2, \tag{4.105b}$$

where ϕ_i has the same requirements as μ_i.

We note that $E_1(x) = \begin{bmatrix} 0 \\ e_1(x) \end{bmatrix}$, since there is no error in $\dot{x}_1$, only in $\dot{x}_2$. Therefore, $E_1(x)$ is in the range of M and thus we can use Corollary 4.3. The Moore-Penrose psuedoinverse of M is $M^\dagger = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}$. Rearranging terms in (4.104), we have

$$\dot{x} = Ax + Bu + \begin{bmatrix} 0 & 0 \\ 0 & 0.5 \end{bmatrix} \left(\begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} E_1(x) + \mu_1 \right) \tag{4.106a}$$

$$y = Cx + \begin{bmatrix} 0.25 & 0 \end{bmatrix} \mu_2. \tag{4.106b}$$

We see, since there is no nonlinearity in (4.106b), that $\phi_2 = \mu_2$ and we need only deal with setting $\phi_1 = M^\dagger E_1(x) + \mu_1$.

$$\phi_1 = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ e_1(x) \end{bmatrix} + \mu_1 \tag{4.107}$$

$$= \begin{bmatrix} 0 \\ 2e_1(x) \end{bmatrix} + \begin{bmatrix} 0 \\ \mu_{12} \end{bmatrix} \tag{4.108}$$

$$= \begin{bmatrix} 0 \\ 2e_1(x) + \mu_{12} \end{bmatrix}. \tag{4.109}$$

As in Section 4.4.1, we will use a noise bound of $S(x_0, \phi) \leq 1$. From (4.58) and (4.59), we know that, if $\|M^\dagger E_1(x)\| \leq \widetilde{M} \leq \sqrt{\delta} = 1$, then $\|x(0)\|_Q^2 + \|\mu_1\| \leq \delta - \widetilde{M} = 1 - \widetilde{M}$ guarantees that $\|M^\dagger E_1(x) + \mu_1\| \leq \sqrt{\delta} = 1$. We know that $\|M^\dagger E_1(x)\| \leq$

$\|M^\dagger\|\|E_1(x)\|$ and $\|M^\dagger\| = 2$. Thus $\|M^\dagger E_1(x)\| \leq 2\|E_1(x)\|$. But, from the previous section, we know that $\|E_1(x)\| \leq 0.16$ since it is the error between x and $\sin x$ for $\|x\| \leq 1$. Therefore, $\|M^\dagger E_1(x)\| \leq 2 * 0.16 = 0.32 = \widetilde{M}$ and we note that $\widetilde{M} = 0.32 \leq \sqrt{\delta} = 1$. Hence, $\|\mu_1\| \leq 1 - 0.32 = 0.68$ guarantees that $\|M^\dagger E_1(x) + \mu_1\| \leq 1$. This means that using a noise bound $S(x_0, \mu) \leq 0.68$ for the nonlinear system and v^* from the linearized system will guarantee separation of the nonlinear output sets for our pendulum example.

4.5 Conclusions

This chapter represents the first attempt at developing the nonlinear theory through the application of known linear theory. We presented two different results to approaching the nonlinear fault detection problem, as well as an application of the results to a nonlinear, oscillating pendulum. In the first result, we showed that by scaling the nonlinear system by a nonzero parameter β and using the strictly proper v from the linearized system, that βv would be proper for the scaled nonlinear problem. This result is important because it guarantees fault detection in the nonlinear system with scaling applied. The key is that although the system is scaled down, we have shown that, before we reduce the linear output sets to the zero set, we will have separated the nonlinear output sets.

The second result gives us situations where the proper v (with no scaling) from the linearized problem will separate the nonlinear output sets, except for possibly tighter noise bounds. This problem requires bounds on the nonlinearity in the system. Using these bounds, we then found the appropriate noise bounds which will guarantee detection in the nonlinear system. It has been shown in [26] that the noise bounds for the nonlinear system are not necessarily smaller than the original linear noise bound.

Finally, we applied both results to a nonlinear oscillating pendulum. After finding the appropriate β by finding the different bounds used in the result, we linearized the system and found a strictly proper v. Using this β and v, we applied βv to the scaled nonlinear problem and found the noise bound which holds to be 0.12788506. Our theorem predicts that using a noise bound of $\beta^2 = 0.08827$ with βv will separate the

nonlinear output sets. We verified that the theorem holds and we note that, although the noise bound that actually holds is much larger than β^2, Theorem 4.2 gives no predictions on how restrictive the bound is. It says merely that β^2 will work for the nonlinear problem.

For the second result, we used Corollary 4.3 to find a noise bound for which the linearized proper signal would be proper for the nonlinear problem. The noise bound was computed to be 0.68. This bound is smaller than the original bound of 1, but is still an acceptable noise bound for most requirements.

Chapter 5

Matlab Conversion

5.1 Conversion Problems

One of our sponsors was NAVSEA in Philadelphia. They have some large systems modeled in Simulink and needed the MEDS algorithm written in the Matlab environment. The goal of this portion of our research was to port a version of continuous.sce, a Scilab code, over to Matlab. There were several issues involved is this process. In particular, the lack of Matlab commands corresponding to the original Scilab commands and the way in which Scilab, versus Matlab, handles passing of arguments during function calls.

First, we will address the lack of similar commands. For the most part, the exact command from Scilab could be found in Matlab. The main exception was the Scilab command $[B, X, bs] = bdiag(A)$. Bdiag takes the matrix A and performs a block diagonalization on it through multiplication by a matrix of its generalized eigenvectors. Using the notation above, we have that $B = X^{-1}AX$, where X is the matrix of eigenvectors. The final ouput, bs, is a vector consisting of the sizes of the blocks in B, i.e. if B_i, the i-th block on the diagonal of B, is 2×2, then $bs(i) = 2$. Until Matlab 7.0, there did not exist a command that was remotely similar to bdiag. When 7.0 came out, they included a new command called $[X, B, bs] = bdschur(A)$. This command also block diagonalizes A by using a similarity transformation with its eigenvectors. However, it computes the eigenvectors in such a way that the blocks on

the diagonal of B are quasi-upper triangular Schur matrices. Also, note the different syntax in using the command. In Scilab, the output is the block diagonal matrix, then the matrix of eigenvectors, where as in Matlab, these two are reversed. Differences in syntax between the two softwares was fairly common, although, for the most part, easily adjusted.

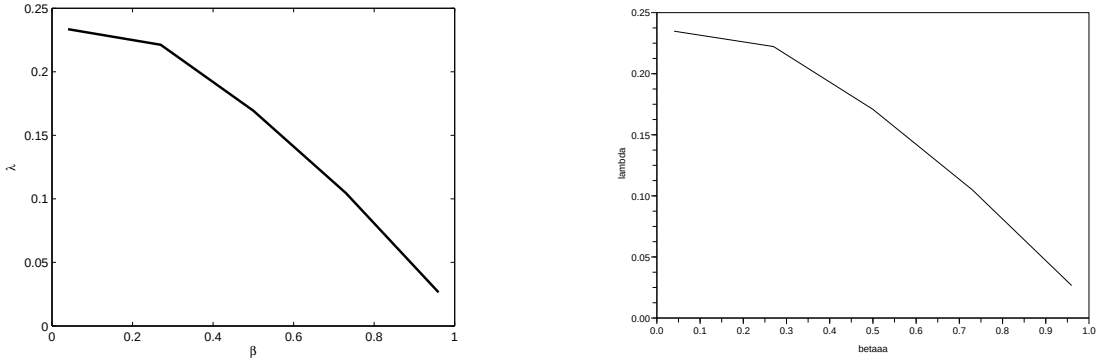
The next problem that arose was the way Scilab handles arguments when calling functions. The code is written with a main script which calls functions that are written above it. In Scilab, if code is written in this way, input arguments do not need to be passed to the function. However, this is not the case in Matlab. In Matlab, the input arguments are considered local inside the script or function and cannot be seen by another function. In most cases, this problem was easily fixed as it was obvious to see which arguments needed to be passed and these were added to the function call. The one case where the syntax of the call was not immediately obvious was inside the function `tpbvs`. `Tpbvs` calls ode45 to solve an ode until it crosses a certain plane. ode45 has a root finding option, which can be used by calling options=odeset('events',@sysr), where @sysr is the call to the function which contains the formula for the plane. Then when calling the ode solver, we would normally just include options as the last parameter of the call. The problem was that we needed to pass arguments to both sysr and sys, the function that contained the ode. When setting ode options, you cannot merely place the input arguments at the end of the call to ode45 as is normally the case. This is also true when setting the options for the ode. To alleviate this problem, we had to use anonymous functions to pass the variables. The functions sysr and sys are written as usual, however the call to these functions is very different. Using the ode45 call as an example, our initial instinct was the call this function as ode45(@sys,[tspan],[i.c.],options,varargin), which did not work. The anonymous function call is ode45(@(t,x) sys(t,x,varargin),[tspan],[i.c.],options). What this call does is set up an anonymous function which has two input variables x and t, and then you are able to send sys the variables and the input arguments. The same procedure was used to send arguments to the root finding function sysr.

5.2 Test Problems

In this section, we will present four test problems, in which we examine the results in Matlab and see how they compare to the Scilab outputs.

5.2.1 Test 1: Case Study

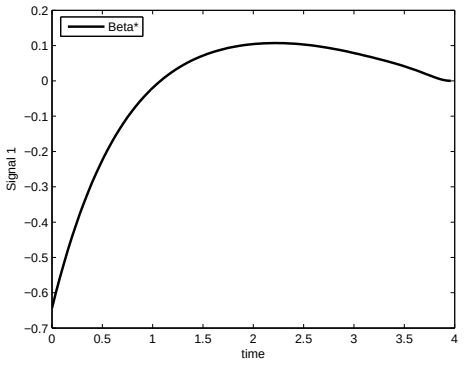
We ran this problem with $n = 5$ and $T = 4$. This was the initial set up which was used during the conversion from Scilab to Matlab. As we can see, figures 5.1(a) and 5.1(b) match up correctly, as do figures 5.2(a) and 5.2(b), and 5.3(a) and 5.3(b). One thing to note is that the Matlab plots for signals one and two originally were the same plot except they were reflected about the t axis. Recall that there are actually two signals, a negative and a positive one. Matlab, in this case, converged to the negative signal while Scilab converged to the positive. In order to show the accuracy of the Matlab code, we plotted the negative of the Matlab plot, making it positive, to make it the same as the Scilab plot for comparison. Table 5.1 gives the results from both Matlab and Scilab. Although some values are different, the are still close enough to be considered valid.



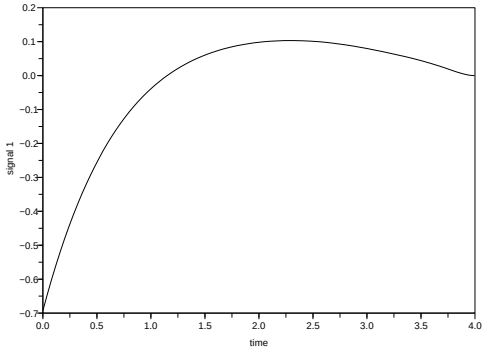
(a) Lambda Beta from Matlab

(b) Lambda Beta from Scilab

Figure 5.1: Lambda Beta graphs from Matlab and Scilab for test one.

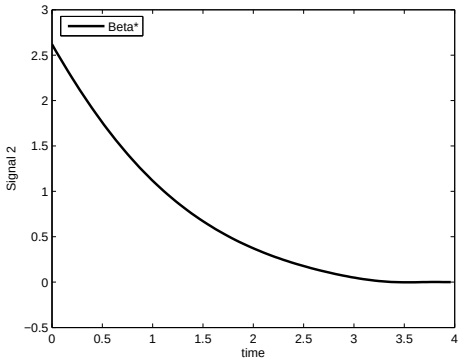


(a) Signal 1 from Matlab

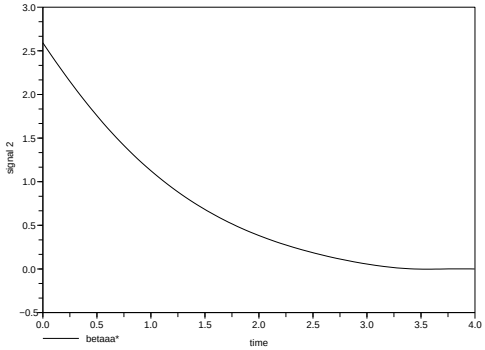


(b) Signal 1 from Scilab

Figure 5.2: Signal one from Matlab and Scilab for test one.



(a) Signal 2 from Matlab



(b) Signal 2 from Scilab

Figure 5.3: Signal two from Matlab and Scilab for test one.

Table 5.1: Results from Matlab and Scilab for case 1.

Code	Time	n	β	λ	Signal 1	Signal 2	Cost
Matlab	[0,1]	5	0.04	0.2335	0.1259	4.1660	4.3052
Scilab	[0,1]	5	0.04	0.2348	0.1457	4.1191	4.2589

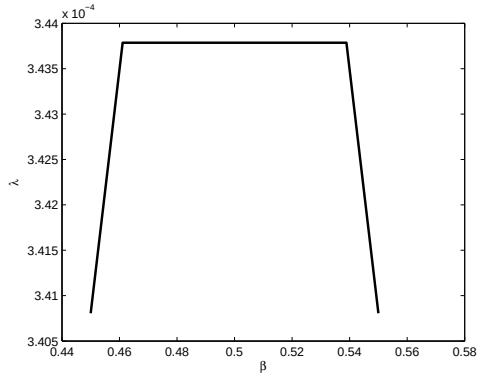
5.2.2 Test Problems 2 and 3: Incipient Paper [1]

Test problems 2 and 3 come from examples used in the paper "Active Incipient Failure Detection: A Nonlinear Case Study". In both problems, we took $n = 10$, and

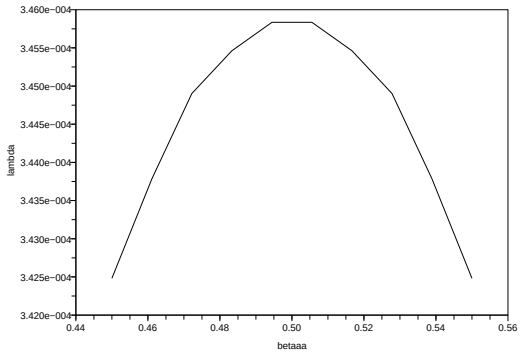
$\delta B = 99B$. We also restricted the interval to run between 0.4 and 0.6 due to the stiffness of the problem. For test case 2, we took $T = 1$ and $V = 1.1$ and for test case 3, we had $T = 10$ and $V = 100$. First, in figure 5.4, we compare the Lambda-Beta graphs. Looking at figures 5.4(a) and 5.4(b) we can see that the time intervals are correct, but that the curve in 5.4(b) is much smoother than the one in 5.4(a). The same lack of smoothness occurs in case 3 as well. While running these particular examples in Scilab, we got convergence errors and had to reset the tolerance and number of steps the integrator should take. Matlab did not produce any errors, so no changes were made to the integrator. If we increased the number of steps and had a tighter tolerance, we would expect the lambda beta graph to smooth out. Figure 5.5 gives us the signal one for both problems. For test one, we had to take the negative of the Matlab signal, since it converged to the negative signal. In both case two and three, Matlab and Scilab both converged to the same signal. As expected, both plots are the same. Figure 5.6 shows the plots of signal 2 and are the same in both tests. Table 5.2 lists the results for both tests and again, although there is a difference in some values, it is insignificant.

Table 5.2: Results for cases 2 and 3.

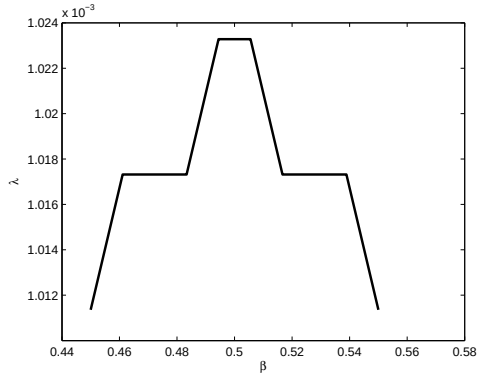
Code	Time	n	β	λ	Signal 1	Signal 2	Cost
Matlab, test 2	[0.4,0.6]	10	0.4611	3.44E-04	2.6817	2.89E+03	2.91E+03
Scilab, test 2	[0.4,0.6]	10	0.4944	0.0003	2.6778	2889.082	2891.5574
Matlab, test3	[0.4,0.6]	10	0.4944	0.001	398.5349	596.4897	1.00E+03
Scilab, test 3	[0.4,0.6]	10	0.4944	0.001	391.8019	586.4112	977.2498



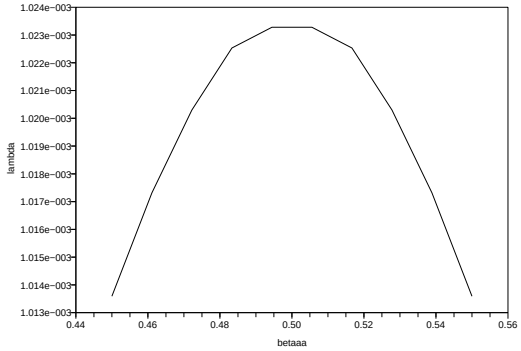
(a) Lambda Beta from Matlab, test 2



(b) Lambda Beta from Scilab, test 2



(c) Lambda Beta from Matlab, test 3

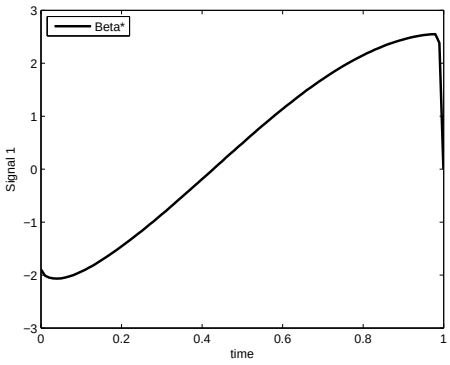


(d) Lambda Beta from Scilab test 3

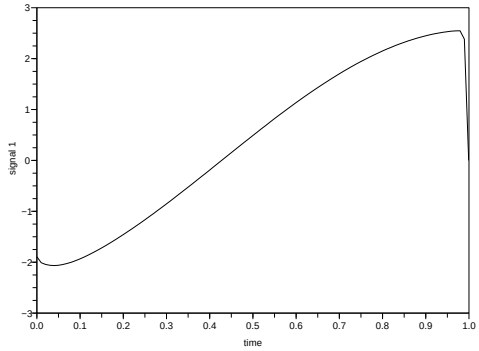
Figure 5.4: Lambda Beta graphs from Matlab and Scilab for tests 2 and 3.

5.2.3 Test Problem 4: Motor Example [2]

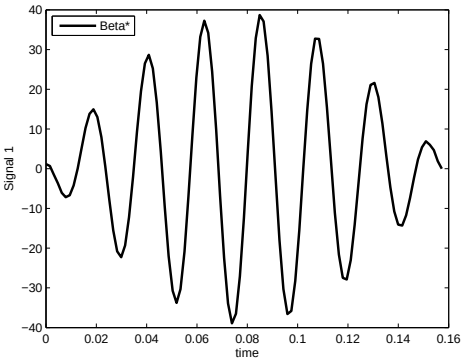
This example was taken from the paper [2]. For this case, we used $n = 10$ and $T = 1$. Figures 5.7(a) and 5.7(b) show the λ_β plots, figures 5.8(a) and 5.8(b) show the signal one plots, and figures 5.9(a) and 5.9(b) give the signal two plots. As expected, all of the plots match. As in case one, the signals in Matlab converged to the negative ones, so we reflected it about the t axis to plot it so that it would match the Scilab output.



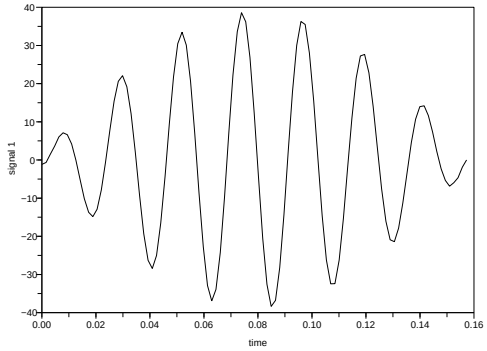
(a) Signal 1 from Matlab, test 2



(b) Signal 1 from Scilab, test 2



(c) Signal 1 from Matlab,test 3

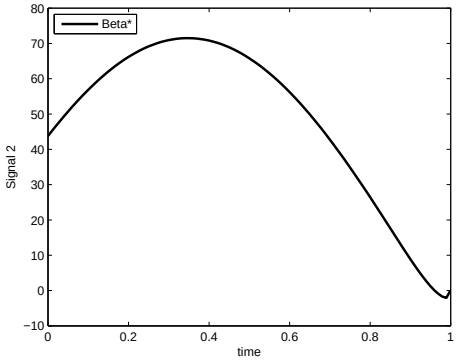


(d) Signal 1 from Scilab, test 3

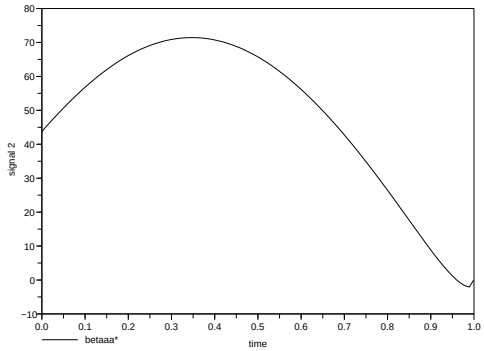
Figure 5.5: Signal one from Matlab and Scilab for tests 2 and 3.

Table 5.3: Results for Matlab and Scilab, test 4.

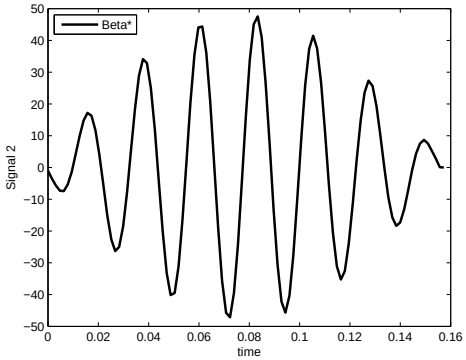
Code	Time	n	β	λ	Signal 1	Signal 2	Cost
Matlab, test 4	[0,1]	10	0.4467	5.84E-05	1.70E+04	1.70E+04	1.69E+04
Scilab, test 4	[0,1]	10	0.4467	0.0001	17115.73	17115.73	17114.8611



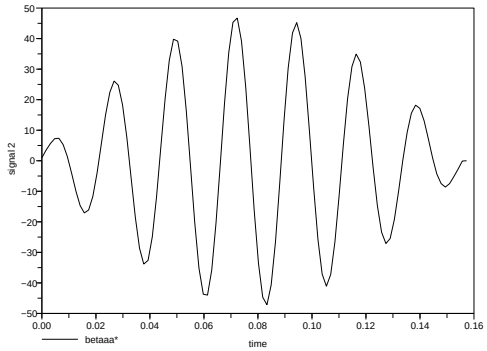
(a) Signal 2 from Matlab, test 2



(b) Signal 2 from Scilab, test 2

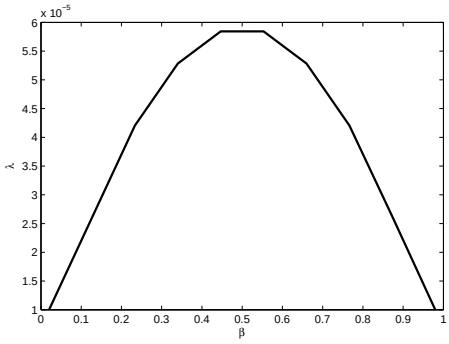


(c) Signal 2 from Matlab, test 3

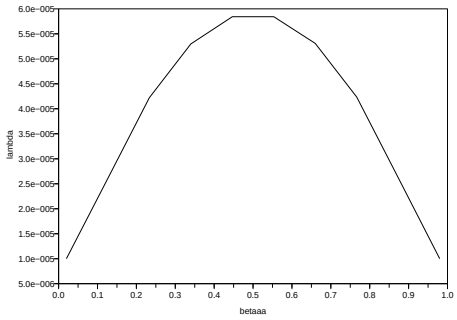


(d) Signal 2 from Scilab, test 3

Figure 5.6: Signal two from Matlab and Scilab for tests 2 and 3.

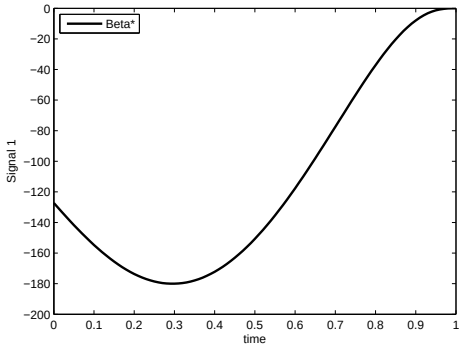


(a) Lambda Beta from Matlab

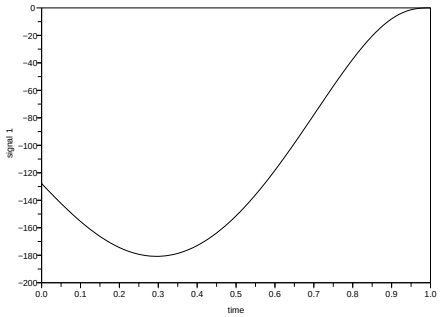


(b) Lambda Beta from Scilab

Figure 5.7: Lambda Beta graphs from Matlab and Scilab for test four.

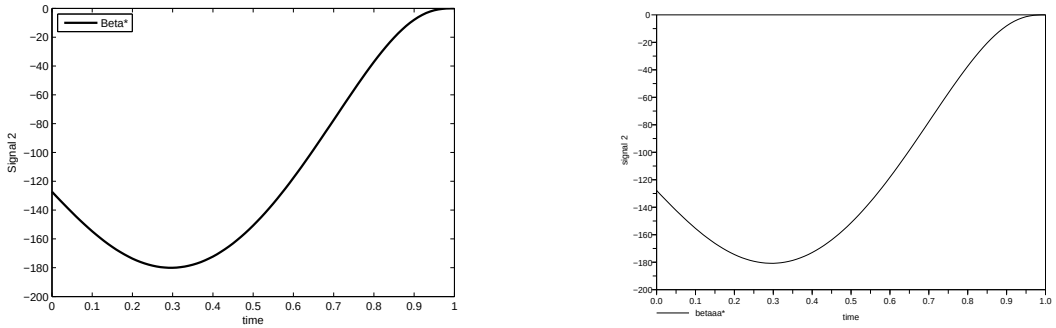


(a) Signal 1 from Matlab



(b) Signal 1 from Scilab

Figure 5.8: Signal one from Matlab and Scilab for test four.



(a) Signal 2 from Matlab

(b) Signal 2 from Scilab

Figure 5.9: Signal two from Matlab and Scilab for test four.

Chapter 6

Conclusions and Contributions

There were several projects which contributed to the completion of this dissertation. The main goal was to research the application of the linear fault detection algorithm to nonlinear problems. As described in the introduction, there are two different approaches to solving the fault detection problem, the Minimum Energy Detection Signal (MEDS) algorithm and the Direct Optimization algorithm. In order to give an overview of the two algorithms, we performed a comparison of the two algorithms to two power systems, as presented in Chapter 2. We showed that although there is only one optimal test signal, there are many "almost" optimal signals depending on the software used. Both softwares produce approximations of the optimal signal, so any of the signals produced are applicable. This comparison lead to the following publication.

- Kimberly J. Drake, Stephen L. Campbell, Ivan V. Andjelkovic, Benjamin L. Hannas, and Kelly A. Sweetingham, *Application of Robust Failure Detection Algorithms to Power Systems*, Proc. 13th Intelligent Systems Applications to Power Systems (ISAP2005), Washington, DC, 2005.

Before starting on the analysis, we performed several case studies to ensure that linearizations were an applicable approach. These case studies were presented in Chapter 3. We first showed that linearizations can produce acceptable signals for the nonlinear problem, even for highly stiff problems. We also showed that linearizations

can work for incipient faults as well. This is important because incipient faults are slight drifts in parameters, thus, the linearization does not lose sensitivity to these drifts. This work produced two publications.

- S.L. Campbell, K.J. Drake, I. Andjelkovic, K. Sweetingham and D. Choe, *Model Based Failure Detection using Test Signals from Linearizations: A Case Study*, Proc IEEE Int. Conf. Control Applications, Munich, 2006.

- K.J. Drake, S.L. Campbell, I. Andjelkovic, and K. Sweetingham, *Active Incipient Failure Detection: A Nonlinear Case Study*, Proc 4th International Conference on Computing, Communications and Control Technologies: CCCT 06, Orlando, 2006.

The main contribution of this dissertation is the nonlinear analysis performed to determine when exactly the linear algorithm can be applied to the nonlinear system, presented in Chapter 4. We first showed that we can parameterize the entire problem by a parameter β and guarantee separation of the nonlinear output sets at some point before the linear output sets are empty. In the second result, we compute the optimal signal for the linearized system, without parameterizing it. We then use this exact signal on the nonlinear system and show how to compute the noise bound which will hold for this system. Finally, a case study using a nonlinear pendulum is presented to verify the results. This research contributed to parts of following publications

- Kimberly J. Drake, Stephen L. Campbell, Ivan Andjelkovic, and Kelly Sweetingham. *Model Based Failure Detection on Nonlinear Systems: Theory and Transition*, Proc ASNE Day 2007, Arlington VA, June 35-36, 2007.

- Kimberly J. Drake, Stephen L. Campbell, Ivan Andjelkovic, and Kelly Sweetingham. *Model Based Failure Detection on Nonlinear Systems: Theory and Transition*, Naval Engineers Journal, Vol. 19, Issue 2, 2007, to appear.

- Ivan Andjelkovic, Kelly Sweetingham and Stephen L. Campbell. *Active Fault Detection in Nonlinear Systems Using Auxiliary Signals*, American Control Conference, June 11, to appear.

and the entire research (two results and case study) can be found in

- Kelly Sweetingham and Stephen L. Campbell. *Auxiliary Signal Design for Fault Detection in Nonlinear Systems*, submitted.

Presentations on this work were given at

- IEEE International Conference on Control Applications, Munich, Germany, October 2006 (invited)
  - Chair for a special session on Process Control, Fault Detection and Tools

Finally, since my research was supported by NAVSEA, part of the terms of funding was to port the MEDS algorithm developed in [11] to Matlab. It was originally produced using Scilab, but NAVSEA uses and prefers Matlab. The resulting software is given in Appendix A. During the process of this conversion, several complications arose. The first one was fairly minor and just involved slight changes in the syntax associated with the different software. The second problem was more complicated. Until Matlab Version 7.0 came out, there was a command used in Scilab that did not exist in Matlab. Once this version came out, the complication was resolved and the port was finished.

The research presented in this dissertation will prove to be important to the fault detection area because it is the first attempt to take the previous developed linear theory and apply it to nonlinear problems. This development is imperative because most real world models are nonlinear by nature and thus, it means there is no need to redevelop all this theory on the nonlinear side or produce any new software, we must only modify the preexisting code.

Chapter 7

Future Work

We have now shown that the MEDS algorithm can be applied to systems with nonlinearities in the state. There are several unanswered questions in this field and a few open problems are described below.

7.1 Nonlinearities

7.1.1 Nonlinearities in the Input Signal

We assumed in Chapter 4 that our system was of the form

$$\dot{x} = f(x) + Bv + M\mu \tag{7.1a}$$

$$y = g(x) + Dv + N\mu. \tag{7.1b}$$

One way of extending this result is to assume nonlinearities in the input signal as well. In this case, our system is

$$\dot{x} = f(x, v) + M\mu \tag{7.2a}$$

$$y = g(x, v) + N\mu. \tag{7.2b}$$

Incorporating this "extra" nonlinearity into the analysis of chapter 4 should not be too difficult. Instead of linearizing around $x = 0$, we now linearize around $v = 0$ as well. Recall that the proof in section 4.2 consisted of four parts, the first two show

linear boundedness, the third shows quadratic boundedness and the final step computes β. For the first two steps, there is no significant difference in the computations when including the input nonlinearity. However, for the third step, we now must include the error in this linearization. This means that $h(x, v) = O(\|\widetilde{x}\|^2) + O(\|v\|^2)$ rather than $h(x) = O(\|\widetilde{x}\|^2)$.

For the second result presented in section 4.3, the adjustment is fairly easy. We treated the error from the linearization of x as part of the noise, so we can also incorporate the error from the linearization of v into the noise as well.

In the nonlinear control literature there is a frequent assumption that the models are affine in the input. This case is intermediate between (7.1) and (7.2)

$$\dot{x} = f(x) + B(x)v + M\mu \tag{7.3a}$$

$$y = g(x) + D(x)v + N\mu. \tag{7.3b}$$

One other case that can be addressed is when the noise is included in the nonlinearity, i.e. we have

$$\dot{x} = f(x, \mu) + Bv \tag{7.4}$$

$$y = g(x, \mu) + Dv. \tag{7.5}$$

An interesting question that arises is whether, in this case, we can use the theory developed for the model uncertainty situation.

7.1.2 Set Point Assumptions

We also made the assumption that our set points were the same (in fact, we linearized around $x = 0$). However, there are many situations where the faulty model does not have the same set point as the properly working model. Another approach to the set point issue is to just compute the set point for one model and use it (even if it's not the correct set point) for the other model. Doing this will add an additional known input to the other model. This type of linear system is studied in [18]. One other consideration is when there are multiple set points and the user must choose between them.

7.2 Discrete Systems

[10] discusses the application of the MEDS algorithm to discrete systems. In fact, an explicit discrete version of the algorithm exists. We would like to be able to extend the results presented in this dissertation from continuous systems to discrete nonlinear ones as well. This process may in fact be easier in the discrete case.

The linear discrete model is of the form

$$x_{k+1} = Ax_k + Bv_k + M\mu_k \tag{7.6a}$$

$$y_k = Cx_k + Du_k + N\mu_k \tag{7.6b}$$

with a noise bound of

$$(x_0 - \bar{x}_0)^T P(x_0 - \bar{x}_0) + \sum_{i=0}^{N-1} |\mu_i|^2 < d \tag{7.7}$$

and the nonlinear model is of the form

$$x_{k+1} = f_k(x_k) + Bv_k + M\mu_k \tag{7.8a}$$

$$y_k = g_k(x_k) + Du_k + N\mu_k \tag{7.8b}$$

7.3 Other Applications

Digital signals for linear, continuous systems has been heavily studied in [31]. It should be asked whether the approaches taken in this paper can be extended to nonlinear continuous systems. Also, the research presented in this thesis dealt only with nice, bounded nonlinearities. Another question that can be asked is what happens with other types of nonlinearities? Nonlinearities can be saturation nonlinearities, or bounded only in some regions. Can the techniques used here be modified to incorporate these types of nonlinearities?

Bibliography

[1] K.J. Drake, S.L. Campbell, I. Andjelkovic, and K. Sweetingham. Active incipient failure detection: A nonlinear case study. *Proc. 4th Int. Conf. on Computing, Communications and Control Technologies (CCCT)*, 2006.

[2] K. Drake, S. Campbell, I. Andejelkovic, B. Hannas, and K. Sweetingham. Applications of robust failure detection algorithms to power systems. *Thirteenth Intelligent Systems Applicatioins to Power Systems Conference*, 2005.

[3] R. Isermann. *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance.* Springer, Germany, 2206.

[4] R. Isermann. Fault diagnosis of machines via parameter estimation and knowledge processing - tutorial paper. *Automatica*, 29, 1993.

[5] M Bisiacco and M.E. Valcher. Observation-based fault detection and isolation for 2d state-space models. *Multidimens. Syst. Signal Process.*, 17:219–242, 2006.

[6] M. Kinnaert, M. Hanus, and Ph. Arte. Fault detection and isolation for unstable linear systems. *IEEE Trans. Automat. Control*, 40:740–742, 1995.

[7] B. Cheung, G. Kumar, and S.A. Rao. Statistical algorithms in fault detection and prediction: Toward a healthier network. *Bell Labs Technical Journal*, 9(4):171–185, 2005.

[8] M.M. Polycarpou, A.T. Vemuri, and A.R. Ciric. Nonlinear fault diagnosis in differential algeraic system. *IEEE Trans. on Systems, Man & Cybernetics: Part A*, 31(2):143–152, 2001.

[9] H. Niemann. A setup for active fault detection. *IEEE Trans. on Automatic Control*, 51(9):1572–1578.

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

[11] D. Choe, S.L. Campbell, and R. Nikoukhah. Auxiliary signal design for robust failure detection: A case study. *Proc. Fifth International Conference on Control and Automation (ICCA)*, 2005.

[12] K. Drake. *Analysis of Numerical Methods for Fault Detection and Model Identification in Linear Systems with Delays.* PhD thesis, North Carolina State University, 2003.

[13] J.T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming.* SIAM, Philadelphia, 2001.

[14] Siso example: the dc motor, matlab control system toolbox help, Version 7.0 (R14).

[15] Control tutorials for matlab, website:
http://www.engin.umich.edu/group/ctm/examples/motor/motorl.html,
created: August 1997, accessed: May 17, 2005.

[16] M. Brain. How gas turbine engines work,
http://science.howstuffworks.com/turbine.htm.

[17] A. Mehta, H. Kaufman, R. Ravi, and K.D. Minto. Continuous time indentification of turbine system models. *Proc. of the 4th IEEE Conference on Control Applications*, pages 412–422, 1995.

[18] R. Nikoukhah and S.L. Campbell. Auxiliary signal design for active failure detection in uncertain linear systems with non-zero initial conditions and known inputs. *Automatica*, 42:219–228, 2006.

[19] S.S. Ke and J.S. Lin. Sensorless speed tracking control with backsetepping design scheme for permanent magnet synchronous motors. *Proc. of the 2005 Conf. on Control App.*, 2005.

[20] J.P. Chancelier, F. Delebecque, C. Gomez, M. Goursat, R. Nikoukhah, and S. Steer. *An Introduction to Scilab.* Springer-Verlag, 2002.

[21] S.L. Campbell, Jean-Philippe Chancelier, and R. Nikoukhah. *Modeling and Simulation in Scilab/Sicos.* Springer, 2005.

[22] S.L. Campbell and R. Nikoukhah. Robust detecction of incipient faults: An active approach. *Proc. 14th Mediterranean Conf. on Control and Automation (MED06)*, 2006.

[23] R. Nikoukhah, S.L. Campbell, and K.J. Drake. An active approach for detection of incipient faults. *International Journal Systems Science*, 2008.

[24] R. Nikoukhah and S.L. Campbell. On the detection of small parameter variations in linear uncertain systems. *European Journal on Control*, to appear.

[25] K. Horton. *Fault Detefction and Model Identification in Linear Dynamical Systems.* PhD thesis, North Carolina State University, 2001.

[26] S.L. Campbell, K.J. Drake, I. Andjelkovic, and K. Sweetingham. Model based failure detection using test signals for linearizations: A case study. *Proc. IEEE Int. Conf. Control Applications*, 2006.

[27] I. Andjelkovic, K. Sweetingham, and S.L. Campbell. Active fault detection in nonlinear systems using auxiliary signals. *Proc. American Control Conference (ACC)*, 2008.

[28] E.D. Sontang. *Mathematical Control Theory.* Springer-Verlag, NY, 1988.

[29] S.L. Campbell and C.D. Jr. Meyer. Generalized inverses of linear transformations. *Surveys and Reference Works in Mathematics*, 1979.

[30] W.E. Boyce and R.C. DiPrima. *Elementary Differential Equations and Boundary Value Problems.* John Wiley & Sons, NY, 1997.

[31] D. Choe. *Digital Signal Design for Fault Detection in Linear, Continuous Dynamical Systems.* PhD thesis, North Carolina State University, 2007.

Appendices

The following Matlab code is the port from SciLab to Matlab discussed in Chapter 5. The original SciLab code can be found in [11].

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% References to equation numbers come from the   %
% paper Auxiliary Signal Design for Robust        %
% Failure Detection: A Case Study, by D. Choe,   %
% S.L. Campbell, and R. Nikoukhah.               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function scilab_conversion
%%%%%%%% MAIN CODE %%%%%%%%%
n=5; %grid on beta interval
d=1/n;
a=d/5;
b=1-d/5; %beta interval
t_from=a-d/5;
t_to=b+d/5;
bb=linspace(a,b,n); %beta grid points
for k=1 %compute suboptimal when k=2, optimal signal k=1
    W_G=0; % Define weights on ICs of cost function (model uncertainty)
    W_H=0;
    J_W=-1; %weight on uncertainty, J_W=1 (model), J_W=-1 (additive)
    % Define A, B, M, C, D, and N for models 0 (unfailed) and 1 (failed)
    A0=[-.1689 .0759 -.9952; -26.859 -2.5472 .0689; 9.3603 -.1773 -2.4792];
    A1=[1 1 0;0 1 1; 0 0 1];
    B0=[zeros(1,2); eye(2,2)];
    B1=B0;
    M0=[eye(3,3) eye(3,3) zeros(3,2)];
    M1=M0;
```

```
27      G0=W_G*[.01 .001 .01; 4.13 .1 .001; 1 .001 .1; zeros(3,3)];
28      G1=W_G*[.01 .1 .01; 0 .1 .1; 0 0 .1; zeros(3,3)];
29      H0=W_H*[zeros(4,2); .1*eye(2,2)];
30      H1=W_H*[zeros(4,2); 11*eye(2,2)];
31      C0=[1 0 0; 0 .9971 .0755];
32      C1=C0;
33      D0=zeros(size(C0*B0));
34      D1=D0;
35      N0=[zeros(2,6) eye(2,2)];
36      N1=N0;
37      P0=eye(size(A0));
38      P1=P0;
39      %computation of sizes
40      nx0=size(A0,2);
41      nx1=size(A1,2);
42      nv=size(B0,2);
43      nnu0=size(M0,2);
44      nnu1=size(M1,2);
45      ny=size(C0,1);
46      nz0=size(G0,1);
47      nz1=size(G1,1);
48      nx=nx0+nx1;
49      nzy0=ny+nz0;
50      nzy1=ny+nz1;
51      %definition of J and P matrices
52      J0=blkdiag(eye(nnu0,nnu0), -J_W*eye(nz0,nz0));
53      J1=blkdiag(eye(nnu1,nnu1), -J_W*eye(nz1,nz1));
54      %sets up info to compute xi
55      nf=nx0;
56      b=0;
57      Om=b*eye(nf,nf);
58      Ga=b*eye(nf,nf);
```

```
59      F=zeros(size(A0));
60      G=zeros(size(B0));
61      T=4; %test period
62      %construct augmented system  (see equations 6,7a,7b)
63      [A,B,C,D,M,N]=datas(A0,B0,C0,D0,G0,H0,M0,N0,A1,B1,C1,D1,G1,H1
            ,M1,N1);
64      %optimization over beta
65      if k==1
66          lb=[];
67          tic;
68          % For all of the betas, compute lambda_beta
69          for beta=bb
70              lb=[lb lbcalc(beta,P0,P1,J0,J1,A,B,C,D,M,N,F,G,Ga,Om,
                    nv,...
71                          nx,ny,nz1,nz0,nf,nnu0,nnu1,T)];
72          end;
73          t_lambda=toc;
74          [lam,i]=max(lb); % find largest lambda
75          beta=bb(i);  % find corresponding beta
76          % Plot the lambda_beta function
77          figure(1)
78          plot(bb,lb,'-k','LineWidth',2);
79          xlabel('\beta');
80          ylabel('\lambda');
81      else
82          betaa=bb(25);
83          lam=lb(25);
84      end
85      %find optimal signals
86      t0=0;
87      grid=101;
88      TT=linspace(t0,T,grid);
```

```
89      tic;
90      [AA,V0,VT,B,C,D,Gx,S,R]=rio(lam,beta,A,B,C,D,F,G,Ga,M,N,P0,P1
            ,Om,J0,J1,...
91      nv,nx,ny,nz1,nz0,nf,nnu0,nnu1);
92      [t,xx]=tpbvs(AA,V0,VT,T,101);
93      nuv=-inv(Gx)*[D'*inv(R)*C,D'*inv(R)*S'-B']*xx;
94      t_signal=toc();
95      t_total=t_lambda+t_signal;
96    % calculate alpha
97    tau=(T-t0)/(grid-1); %uniform time interval
98    cost=delta(xx,TT,Ga,nx,nf,nv,tau,tf,nuv,Om,F,G,T);
99    alpha=1/(sqrt(cost*lam));
100   % calculate cost=delta(v)
101   [nrnuv,ncnuv]=size(nuv);
102   nuv(nrnuv-nv+1:nrnuv,:)=alpha*nuv(nrnuv-nv+1:nrnuv,:); %must
          update nuv!!
103   cost=delta(xx,TT,Ga,nx,nf,nv,tau,tf,nuv,Om,F,G,T);
104   signal_1=tau*nuv(nrnuv-nv+1,1:ncnuv-1)*nuv(nrnuv-nv+1,1:ncnuv
          -1)';
105   signal_2=tau*nuv(nrnuv,1:ncnuv-1)*nuv(nrnuv,1:ncnuv-1)';
106   %%%%%%%% BEGIN PLOTS %%%%%%%%
107   %figures (2),(3)=signals 1 and 2
108   figure(2)
109   plot(t',-nuv(nrnuv-nv+1,:)','-k','LineWidth',2)
110   xlabel('time')
111   ylabel('Signal 1')
112   legend('Beta*',2)
113   figure(3)
114   plot(t',-nuv(nrnuv,:)','-k','LineWidth',2)
115   xlabel('time')
116   ylabel('Signal 2')
117   legend('Beta*',2)
```

```
118 %   %figures (4),(5) scaled signals 1 and 2
119 %   figure(4)
120 %   plot(t',nuv(nrnuv-nv+1,:)'/nuv(nrnuv-nv+1,1),'-k','LineWidth
        ',2)
121 %   xlabel('time')
122 %   ylabel('Scaled Signal 1')
123 %   legend('Beta*',2)
124 %
125 %   figure(5)
126 %   plot(t',nuv(nrnuv,:)'/nuv(nrnuv,1),'-k','LineWidth',2)
127 %   xlabel('time')
128 %   ylabel('Scaled Signal 2')
129 %   legend('Beta*',2)
130 %
131 %   %figures (6),(7),(8) plot x_j(t)
132 %   figure(6)
133 %   plot(t',xx(nx+1,:)','-k','LineWidth',2)
134 %   xlabel('time')
135 %   ylabel('xi(i)')
136 %   legend('Beta*',2)
137 %
138 %   figure(7)
139 %   plot(t',xx(nx+nf-1,:)','-k','LineWidth',2)
140 %   xlabel('time')
141 %   ylabel('xi(2)')
142 %   legend('Beta*',2)
143 %
144 %   figure(8)
145 %   plot(t',xx(nx+nf,:),'-k','LineWidth',2)
146 %   xlabel('time')
147 %   ylabel('xi(3)')
148 %   legend('Beta*',2)
```

```
149 end
150  %outputs the results
151  W_G,W_H,time=[t_from,t_to],n,beta,lam,signal_1,signal_2,cost
152 end
153 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% END MAIN CODE
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
154 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% BEGIN FUNCTIONS
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
155 function [A,B,C,D,M,N]=datas(A0,B0,C0,D0,G0,H0,M0,N0,A1,B1,C1,D1,
        G1,H1,M1,N1)
156 %This function builds the augmented matrices which combine
157 %the failed and unfailed models into one system (see equations
        6,7,7b)
158       [ny,nnu0]=size(N0);
159       [ny,nnu1]=size(N1);
160       [nz1,nx1]=size(G1);
161       [nz0,nx0]=size(G0);
162       E0=[zeros(nz0,ny); eye(ny,ny)];
163       E1=[zeros(nz1,ny); eye(ny,ny)];
164       M0=[M0 zeros(nx0,nz0)];
165       M1=[M1 zeros(nx1,nz1)];
166       N0=[zeros(nz0,nnu0) -eye(nz0,nz0); N0 zeros(ny,nz0)];
167       N1=[zeros(nz1,nnu1) -eye(nz1,nz1); N1 zeros(ny,nz1)];
168       C0=[G0; C0];
169       C1=[G1; C1];
170       D0=[H0; D0];
171       D1=[H1; D1];
172       F=null([E0; E1]')';
173       F0=F(:, 1:nz0+ny);
174       f2=size(F,2);
175       F1=F(:,1+nz0+ny:f2);
176       %Final augmented system matrices
```

```
177       B=[B0; B1];
178       A=blkdiag(A0,A1);
179       M=blkdiag(M0,M1);
180       C=[F0*C0 F1*C1];
181       D=F0*D0+F1*D1;
182       N=[F0*N0 F1*N1];
183    end
184    function lam=lbcalc(beta,P0,P1,J0,J1,A,B,C,D,M,N,F,G,Ga,Om,nv,nx
          ,...
185       ny,nz1,nz0,nf,nnu0,nnu1,T,prec)
186    % This function computes optimal lambda for a given beta
187       sprintf('Computing lambda for beta=%f\n',beta)
188       rhs=nargin;       % number of inputs (rhs) and outputs (lhs)
189       lhs=nargout;
190       if rhs==24
191          prec=.01;
192       end
193       lmin=0.00001;    % Set up min and max for lambda
194       lmax=100;
195       [AA,V0,VT]=rio(lmax,beta,A,B,C,D,F,G,Ga,M,N,P0,P1,Om,J0,J1
             ,...
196          nv,nx,ny,nz1,nz0,nf,nnu0,nnu1);    %calls function rio
197       t=tpbvs(AA,V0,VT,T);  %calls function tpbvs
198       if t<T
199          lam=0;
200          sprintf('does not work');
201          return;
202       end
203       while lmax-lmin>prec*lmin  %loop to optimize lambda for given
             beta
204          lam=(lmax+lmin)/2;
```

```
205             [AA,V0,VT]=rio(lam,beta,A,B,C,D,F,G,Ga,M,N,P0,P1,Om,J0,J1
                    ,...
206             nv,nx,ny,nz1,nz0,nf,nnu0,nnu1);  %calls function rio
207             t=tpbvs(AA,V0,VT,T);      %calls function tpbvs
208             if t<T
209                 lmin=lam;
210             else
211                 lmax=lam;
212             end
213         end
214         lam=lmin;  %output lambda
215     end
216     function [AA,V0,VT,B,C,D,Gx,S,R]=rio(lam,beta,A,B,C,D,F,G,Ga,M,N,
            P0,P1,Om,J0,J1,...
217             nv,nx,ny,nz1,nz0,nf,nnu0,nnu1)
218             %Sets up the matrices to be used in the solving the
219             %two point boundary value system (see eqs. 10,11)
220         Pb=blkdiag(1/beta*P0, 1/(1-beta)*P1);
221         Jb=blkdiag(beta*J0, (1-beta)*J1);
222         T=[M B; N D]*inv(blkdiag(Jb, -lam*eye(nv,nv)))*[M B; N D]';
223         Q=T(1:nx,1:nx);
224         nt2=size(T,2);
225         S=T(1:nx,nx+1:nt2);
226         nt1=size(T,1);
227         R=T(nx+1:nt1, nx+1:nt2);
228         No=null(N);
229         if min(real(eig(No'*Jb*No)))<=0
230             error('test 1 failed');
231         end
232         D=[N D];
233         C=[C zeros(ny+nz1+nz0,nf)];
234         A=blkdiag(A,F);
```

```
235       B=[M B; zeros(nf,nnu0+nnu1+nz0+nz1) G];
236       Gx=blkdiag(Jb, -lam*eye(nv,nv));
237       T=[B; D]*inv(Gx)*[B; D]';
238       [ntr,ntc]=size(T);
239       Q=T(1:nx+nf,1:nx+nf);
240       S=T(1:nx+nf,nx+nf+1:ntc);
241       R=T(nx+nf+1:ntr,nx+nf+1:ntc);
242       AA=[A-S*inv(R)*C Q-S*inv(R)*S'; C'*inv(R)*C+blkdiag(zeros(nx,
              nx), -lam*Ga) -(A-S*inv(R)*C)'];
243       V0=[inv(Pb) zeros(nx,nf) -eye(nx,nx) zeros(nx,nf); zeros(nf,
              nx) eye(nf,nf) zeros(nf,nx+nf);
244            zeros(nx+nf,2*nx+2*nf)];
245       VT=[zeros(nx+nf,2*nx+2*nf); zeros(nx,nx+nf) eye(nx,nx) zeros(
              nx,nf);
246            zeros(nf,nx) -lam*Om zeros(nf,nx) eye(nf,nf)];
247  end
248  function phd=sys(t,Ph,h,n,Ab,Af,IIJJr)
249  %ode for tpbvs (see equations 14,15)
250       phb=reshape(Ph(1:h^2),h,h);
251       phf=reshape(Ph(h^2+1:IIJJr),n-h,n-h);
252       phbd=-Ab*phb;
253       phfd=Af*phf;
254       phd=[phbd(:); phfd(:)];
255  end
256  function xd=sysb(t,x,Ab)
257       xd=Ab*x;
258  end
259  function Xf=sysf(t,x,Af)
260       Xf=Af*x;
261  end
262  function [r,isterminal,direction]=sysr(t,Ph,h,IIJJr,n,Ab,Af,V0,VT
         )
```

```
263 %builds the surface to check for a crossing (see equation 16)
264     phb=reshape(Ph(1:h^2),h,h);
265     phf=reshape(Ph(h^2+1:IIJJr),n-h,n-h);
266     % r is surface to be checked
267     r=det([blkdiag(eye(size(Ab)),-phf) blkdiag(-phb,eye(size(Af))
        ); V0 VT]);
268     isterminal=1;  %tells the integrator to stop after root found
269     direction=0;    %tells the integrator that the solution can
        either
270                     %increase or decrease
271 end
272 function [T,xx]=tpbvs(A,V0,VT,T,nb)
273 %Solves the two point boundary value problem
274 %(see equation 15)
275     if nargout > 1
276         T=T+2;
277     end
278     [X,A,bs]=bdschur(A);  %block diagonalizes A
279     j=1;
280     sp=[];
281     for i=bs'
282         sp=[sp min(real(eig(A(j:j+i-1,j:j+i-1))))*ones(1,i)];
283         j=j+i;
284     end
285     n=size(A,2);
286     [sp,I]=sort(sp,'descend');
287     A=A(I,I);
288     X=X(1:n,I);
289     V0=V0*X;
290     VT=VT*X;
291     h=max(find(~(sp<0)));
292     Ab=A(1:h,1:h);
```

```
293     [ar,ac]=size(A);
294     Af=A(h+1:ar,h+1:ac);
295     II=eye(size(Ab));
296     JJ=eye(size(Af));
297     [IIJJr,IIJJc]=size([II(:);JJ(:)]);
298     options=odeset('events',@(t,Ph) sysr(t,Ph,h,IIJJr,n,Ab,Af,V0,
            VT));
299     %sets ode options to find a zero crossing of the function
            sysr
300     [t,PPh,rd,ye,ie]= ode45(@(t,Ph) sys(t,Ph,h,n,Ab,Af,IIJJr),[0
            T],[II(:);JJ(:)],options);
301     %solves equation 15
302     if isempty(rd)==0   %sets the new time to be the crossing
            time if a crossing occurs
303         T=rd(1);
304     end
305     if nargout>1
306         if nargin<5
307             nb=100;
308         end
309         [PPhr,PPhc]=size(PPh);
310         phb=reshape(PPh(PPhr,1:h^2),h,h);
311         phf=reshape(PPh(PPhr,h^2+1:PPhc),n-h,n-h);
312         x0T=null([blkdiag(eye(size(Ab)),-phf) blkdiag(-phb,eye(
                size(Af))); V0,VT]);
313         TT=linspace(0,T,nb);
314         xbT=x0T(n+1:n+h);
315         xf0=x0T(h+1:n);
316         [TTr,TTc]=size(TT);
317         k=TT(TTc:-1:1);
318         %solves equations 17a and 17b
```

```
319             [t,Xb]=ode45(@sysb,-TT,xbT,[],Ab); %used -TT to integrate
                    backward in time
320             [t,Xf]=ode45(@(t,x) sysf(t,x,Af),TT,xf0);
321             Xb=Xb';
322             [Xbr,Xbc]=size(Xb);
323             xx=X*[Xb(:,Xbc:-1:1); Xf'];
324             T=TT;
325         end
326     end
327     function cost=delta(xx,TT,Ga,nx,nf,nv,tau,tf,nuv,Om,F,G,T)
328     %a nested function which calculates the cost (see eq 3a)
329     %the ode calls the function fV, which in turn calls v
330         [t,cst]=ode45(@(t,V) fV(t,V,xx,TT,Ga,nx,nf,nv,tau,tf,nuv,Om,F
                ,G,T),...
331             TT,zeros(1+nf,1));
332         cstp=cst';
333         [nrcst,nccst]=size(cstp);
334         [nrxx,ncxx]=size(xx);
335         cost=cstp(1,nccst)+xx(nx+1:nx+nf,ncxx)'*Ga*xx(nx+1:nx+nf,ncxx
                );
336             function Vdot=fV(t,V,xx,TT,Ga,nx,nf,nv,tau,tf,nuv,Om,F,G,
                    T)
337                 %this function is the integral in equation 3a
338                 %requires a call to v
339                 z=V(1,1);
340                 xi=V(2:1+nf,1);
341                 vt=v(t,xx,TT,Ga,nx,nf,nv,tau,tf,nuv,Om,F,G,T);
342                 zdot=vt'*vt+xi'*Om*xi;
343                 xidot=F*xi+G*vt;
344                 Vdot=[zdot;xidot];
345                 function vt=v(t,xx,TT,Ga,nx,nf,nv,tau,tf,nuv,Om,F,G,T
                        )
```

```
346                     %sets up the |v|^2 from equation 3a
347                     i_cur=fix(t/tau)+1;
348                     [nrnuv,ncnuv]=size(nuv);
349                     if t<T-tau
350                         vt=(1-t+TT(i_cur))*nuv(nrnuv-nv+1:nrnuv,i_cur
                                ) ...
351                         +(t-TT(i_cur))*nuv(nrnuv-nv+1:nrnuv,i_cur+1);
352                     else
353                         vt=nuv(nrnuv-nv+1:nrnuv,ncnuv);
354                     end
355                 end
356             end
357 end
```