Abstract

Horton, Kirk Gerritt. Fault Detection and Model Identification in Linear Dynamical Systems. (Under the direction of Dr. Stephen La Vern Campbell.)

Linear dynamical systems, $Ex' + Fx = f(t)$, in which E is singular, are useful in a wide variety of applications. Because of this wide spread applicability, much research has been done recently to develop theory for the design of linear dynamical systems. A key aspect of system design is fault detection and isolation (FDI). One avenue of FDI is via the multi-model approach, in which the parameters of the nominal, unfailed model of the system are known, as well as the parameters of one or more fault models. The design goal is to obtain an indicator for when a fault has occurred, and, when more than one type is possible, which type of fault it is. A choice that must be made in the system design is how to model noise. One way is as a bounded energy signal. This approach places very few restrictions on the types of noisy systems which can be addressed, requiring no complex modeling requirement.

This thesis applies the multi-model approach to FDI in linear dynamical systems, modeling noise as bounded energy signals. A complete algorithm is developed, requiring very little on-line computation, with which nearly perfect fault detection and isolation over a finite horizon is attained. The algorithm applies techniques to convert complex system relationships into necessary and sufficient conditions for the solutions to optimal control problems. The first such problem provides the fault indicator via the minimum energy detection signal, while the second problem provides for fault isolation via the separating hyperplane. The algorithm is implemented and tested on a suite of examples in commercial optimization software. The algorithm is shown to have promise in nonlinear problems, time varying problems, and certain types of linear problems for which existing theory is not suitable.

FAULT DETECTION AND MODEL IDENTIFICATION IN LINEAR DYNAMICAL SYSTEMS

BY

KIRK GERRITT HORTON

A DISSERTATION SUBMITTED TO THE GRADUATE FACULTY OF

NORTH CAROLINA STATE UNIVERSITY

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

OPERATIONS RESEARCH PROGRAM

RALEIGH, NORTH CAROLINA

FEBRUARY 2001

APPROVED BY:

S. L. CAMPBELL

CHAIR OF ADVISORY COMMITTEE

R. SMITH

K. ITO

H. T. TRAN

E. CHUKWU

Biography

Kirk Gerritt Horton was born September 14, 1963 in Paterson, New Jersey. He grew up attending public schools with his three sisters in West Milford, New Jersey. He was the valedictorian of the West Milford High School graduating class of 1981.

He received his B. Engineering degree with a major in Electrical Engineering and Computer Science from Stevens Institute of Technology in Hoboken, New Jersey in 1985. Shortly after graduation he received a commission in the United States Air Force from AFROTC. From 1985 to 1993 he was a pilot, flying the T-37 and the T-38 trainer, then the RF-4C reconnaissance/fighter, and then the F-111E fighter/bomber. He flew 19 missions over enemy territory in Iraq during Operation Desert Storm, receiving the AF Distinguished Flying Cross for one of those missions. From 1993 to 1995 he attended the Air Force Institute of Technology, earning a M. S. in Operations Analysis. From 1995 to 1997 he piloted the F-117A Stealth Fighter, ending that tour as an instructor in the aircraft. He arrived at N. C. State in 1997 to pursue a Ph. D. in Operations Research. He is currently serving on active duty as a Major in the Air Force.

The author is married to the former Susan Elaine Pringels, of Sumter, S. C. and they have two daughters, Madisyn and Ashlyn.

Acknowledgements

As countless students have before me, I owe an incredible amount of gratitude to my advisor, Dr. Stephen L. Campbell. An exceptional teacher, a meticulous researcher, a prolific writer, an accomplished artist, and a natural conversationalist, he guided me through the morass of graduate study with a firm but gentle hand. Without his expertise I would not have been able to complete this project.

I am also grateful to Dr. Ralph Smith, Dr. Hien T. Tran, Dr. Kazufumi Ito, and Dr. Ethelbert Chukwu for serving on my committee. Their mathematical and Operations Research experience were invaluable in ensuring the accuracy of my research.

I would like to thank the United States Air Force, and in particular the faculty and staff at the Air Force Institute of Technology for preparing and selecting me for the program that allowed me to attend North Carolina State University to pursue my degree.

Finally, I must acknowledge my wife, Susan, and my daughters, Madisyn and Ashlyn. Without their love and support none of this would have been possible, nor would it have been worth doing. I pray that all of our father's blessings will be poured out on them as we travel around the world serving our country and spreading the Good News.

Table of Contents

List of Tables vii

List of Figures ix

1 Introduction and Review of Prior Research 1

1.1 Linear Descriptor Systems 2

1.1.1 Basic Theory 2

1.1.2 Numerical Solutions 7

1.2 Fault Detection and Isolation 11

1.2.1 Basic Theory 11

1.2.2 Feedback and Observer Design 13

1.2.3 Optimal Control 16

1.2.4 H_∞ Control 18

1.2.5 Prior Research 19

1.2.6 Conclusion 24

1.3 Outline of Thesis 24

1.4 Contributions of Thesis 25

2 Fault Detection via the Detection Signal **26**

2.1 The Problem - Finding the Minimum Energy Detection Signal 26

2.1.1 Problem Setup 27

2.1.2 Formulation as an Optimal Control Problem 30

2.1.3 Problem Statement 37

2.2 Necessary Conditions 37

2.2.1 Computing the Necessary Conditions 38

2.2.2 Riccati Form of Necessary Conditions 39

2.2.3 Problem Formulation in Terms of the Necessary Conditions . . 49

2.2.4 Sufficient Conditions 52

2.3 The Minimum Energy Detection Signal Algorithm 53

2.4 Variations 58
2.4.1 Multiple Fault Models 59
2.4.2 Unreduced Model 61
2.4.3 Controlled Systems 63
2.4.4 Alternative Cost Functions 64
2.4.5 Knowledge of Initial Conditions 65
2.4.6 Conclusion 66

3 Model Identification via the Separating Hyperplane 68
3.1 The Problem - Determining the Origin of a Given Output 68
3.1.1 Problem Setup 69
3.1.2 The Separating Hyperplane 71
3.1.3 Approximating the Separating Hyperplane 72
3.1.4 Problem Statement 74
3.2 The Model Identification Algorithm 76
3.3 Variations 79
3.3.1 Multiple fault models 79
3.3.2 Alternative Formulations 80
3.3.3 Controlled Systems 81
3.3.4 Alternative Cost Functions 81
3.3.5 Knowledge of Initial Conditions 81
3.3.6 Conclusion 82

4 Examples and Analysis of Results 83
4.1 The Complete Problem and Algorithm 83
4.2 Introduction of Software 87
4.2.1 SOCS Parameters 88
4.2.2 Choosing a Value of ϵ 90
4.3 Introduction of Examples 90
4.4 One-Dimensional State Examples 92
4.4.1 Primary One-Dimensional Example 93
4.4.2 Other One-Dimensional Examples 99
4.5 Two-Dimensional State Examples 103
4.5.1 Primary Two-Dimensional Example 104
4.5.2 Other Two-Dimensional Examples 108
4.5.3 Common Mode Two-Dimensional Example 115
4.6 Industrial Example 118
4.7 Multiple Fault Model Examples 121
4.7.1 One-Dimensional Example 121
4.7.2 Two-Dimensional Example 124

4.8 Conclusion 125

5 Future Work and Conclusions 129

5.1 Future Work 129

5.1.1 The Half-Infinite Interval 130

5.1.2 Linear Time Varying Models 131

5.1.3 Nonlinear Models 132

5.1.4 Independent Noise Bounds 135

5.1.5 Sensitivity Issues 135

5.2 Conclusions 136

List of References 139

A Software Drivers 145

A.1 Model Reduction 145

A.2 Fortran Code Generation 148

A.3 Optimization via the FDMI Algorithm 149

A.4 Analysis and Presentation of Results 163

A.4.1 Detection Signal Phase Processing 163

A.4.2 Model Identification Phase Processing 164

List of Tables

4.1 γ^* and $\|\widehat{v}\|$ for Example 4.1: $t_f = 1, 10, 20, 100$ 95
4.2 Formulation comparison of Example 4.1: $t_f = 1, 10, 20$ 98
4.3 γ^* and $\|\widehat{v}\|$ for Examples 4.1-4.4: $t_f = 1, 10$ 101
4.4 Performance comparison of Example 4.5 on various time intervals . . 107
4.5 Performance comparison of Examples 4.2-4.10: $t_f = 1$ 114
4.6 Performance comparison of Examples 4.2-4.10: $t_f = 10$ 114

List of Figures

3.1 Output sets under application of $\widehat{v}$ and $\delta\widehat{v}$, $\delta > 1$ 73
3.2 Output sets under full and reduced noise contributions 75

4.1 Typical variation of $a_\epsilon(t)$ with ϵ 91
4.2 $\widehat{v}$, for Example 4.1: $t_f = 1$ (left), $t_f = 10$ (right) 93
4.3 $\widehat{v}$ for Example 4.1: $t_f = 20$ (left), $t_f = 100$ (right) 94
4.4 $\widehat{v}$ for Example 4.1: $t_f = 20, 100$ 94
4.5 γ^* for Example 4.1 as a function of t_f 95
4.6 Rescaled $\widehat{v}$ for Example 4.1: $t_f = 1, 10, 20, 100$ 96
4.7 $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.1: $\epsilon = 0.3, 0.5, 0.7, 0.9$ 97
4.8 $\widehat{v}$ for Examples 4.2 (left), 4.3 (center), 4.4 (right): $t_f = 1$ 101
4.9 $\widehat{v}$ for Examples 4.1-4.4: $t_f = 1$ 102
4.10 $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.4: $\epsilon = 0.3, 0.5, 0.7, 0.9$ 103
4.11 $\widehat{v}$ for Example 4.5: $t_f = 1$ (left) and $t_f = 20$ (right) 105
4.12 $\widehat{v}$ for Example 4.5: $t_f = 1, 2, 4, 6, 8, 10, 20$ 105
4.13 γ^* for Example 4.5 as a function of t_f (left) and compared with Example 4.1 (right) 107
4.14 $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.5: $\epsilon = 0.3, 0.5, 0.7, 0.9$ 108
4.15 $\widehat{v}$ for Examples 4.5-4.10: $t_f = 1$ (left), $t_f = 10$ (right) 112
4.16 $\widehat{v}$ for Examples 4.2,4.3,4.4,4.9: $t_f = 10$ 113
4.17 Normalized $\widehat{v}$ for Examples 4.5-4.10: $t_f = 10$ 113
4.18 $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.10: $\epsilon = 0.7$ 115
4.19 Components of $\widehat{v}$ for Example 4.11: $t_f = 5$ 117
4.20 Components of $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.11: $\epsilon = 0.7$ 117
4.21 Components of $\widehat{v}$ for Example 4.12: $t_f = 1$ 120
4.22 Components of $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.12: $\epsilon = 0.7$ 120
4.23 $\widehat{v}$ for Example 4.13 sequential vs. simultaneous (left) and full interval two-model vs. simultaneous (right) 123
4.24 $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.13: sequential solve 123
4.25 $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.13: simultaneous solve 124

4.26 $\widehat{v}$ for Example 4.14 sequential vs. simultaneous (left) and full interval two-model vs. simultaneous (right) 126
4.27 $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.14: sequential 126
4.28 $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.14: simultaneous 127

Chapter 1

Introduction and Review of Prior Research

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

$$Ez' + Fz = f(t) \tag{1.1}$$

in which the (square) matrix E is singular, are called linear descriptor systems. Many systems throughout a wide variety of applications are most easily described as linear descriptor systems. Variational problems subject to constraints, such as the equations of motion for a robotic arm, can often be written as descriptor systems. Network modeling problems, as in electrical circuit design, are another example. The list continues with model reduction problems, singular perturbations, and discretizations of partial differential equations, just to name a few. (See [5, 8] for an in-depth description of applications and examples.) Because of this wide spread applicability, much research has been done recently involving linear DAEs.

A key aspect of system design in linear DAE modeled systems is fault detection and isolation (FDI). One avenue of FDI is via the multi-model approach, in which the parameters of the nominal, unfailed model of the system are known, as well as the parameters of one or more fault models. The design goal is to obtain an indicator that tells the operator when a fault has occurred, and, when more than one type is possible, which type of fault it is.

Another aspect of system design is the modeling of noise. One way to model noise is as a bounded energy signal. This approach places very few restrictions on the types of noisy systems which can be addressed. It also presents no complex modeling requirement, a very useful computational tool of which we can take full advantage.

In this thesis we apply the multi-model approach to FDI in linear descriptor systems, modeling noise as bounded energy signals. The combination appears to be under-explored, in that very little research seems to exist that uses both the multi-model approach and bounded energy noise. We develop a complete algorithm, requiring very little on-line computation by an operator, with which nearly perfect fault detection and isolation over a finite horizon is attained.

1.1 Linear Descriptor Systems

We begin with a short introduction to descriptor systems, the basic theory and several numerical methods used to obtain solutions.

1.1.1 Basic Theory

As described above, DAEs occur in many applications. Models that consist of a set of ordinary differential equations (ODEs) often are first written as DAEs. A DAE is manipulated through differentiation and substitution to convert it to ODE form. Consider

$$x' = ax + by \tag{1.2a}$$

$$y = cx + d \tag{1.2b}$$

in which a, b, c, and d are scalar constants. Equation (1.2) consists of a differential equation (1.2a) and an algebraic constraint (1.2b). The Jacobian of (1.2) with respect to x', y' is

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

which is singular. By differentiating (1.2b) we obtain the full system

$$\begin{aligned} x' &= ax + by && (1.3a) \\ y &= cx + d && (1.3b) \\ y' &= cx'. && (1.3c) \end{aligned}$$

We can substitute (1.3b) into (1.3a) to obtain the ODE

$$\begin{aligned} x' &= (a + bc)x + bd && (1.4a) \\ y' &= cx' && (1.4b) \end{aligned}$$

the solution of which is easily obtained.

Frequently, reasons exist for not attempting to manipulate a system like (1.2) into explicit (ODE) form. First, physical problems initially modeled as DAEs contain relationships between variables of interest. Changing to an explicit model may result in less meaningful variables, as well as a loss of the importance of the relationships between those variables. In addition, sparsity is usually lost. Numerical methods that rely on the sparsity of a DAE may not be suitable for solving the ODE that is obtained by differentiation and substitution. Finally, it may not be easy or even possible to convert a complex system into ODE form. When it is possible, it might be easier to solve the DAE directly than to do the mathematical manipulation necessary to convert it to an ODE. See [5] for a more detailed explanation as to why the DAE form of a model may be preferred over the ODE form.

It is due to these reasons, among others, that the base of research in DAEs has continuously grown over the last several years. At the heart of the theory are two key concepts, *solvability* and the *uniform differentiation index* [5].

DEFINITION 1.1. *The system (1.1), where E and F are $m \times m$ matrices, is* solvable *on an interval if for every m-times differentiable $f(t)$, there is at least one continuously differentiable solution to (1.1). In addition, solutions are defined on the entire interval and are uniquely determined by their value at any t in the interval.*

We will return to the necessary and sufficient conditions for solvability of certain types of DAEs a bit later.

DEFINITION 1.2. *The minimum number of times that all or part of (1.1) must be differentiated with respect to t in order to determine z' as a continuous function of z, t is the* index*, ν, of DAE (1.1).*

Example (1.2) is an index one DAE. Numerical methods are well developed for index one DAEs. Higher index problems are notoriously more difficult to solve via numerical methods. Fortunately, all of the DAEs we will deal with in this thesis are of index one.

Of the several special structural forms for DAEs found in the literature, two will be of interest in this research:

- Linear Time Invariant DAE

$$Ez' + Fz = f(t) \tag{1.5}$$

- Linear Time Varying DAE

$$E(t)z' + F(t)z = f(t) \tag{1.6}$$

We mention three other types for completeness:

- Linear in the derivative, nonlinear DAE

$$E(z)z' + F(z)z = f(t) \tag{1.7}$$

- Semi-Explicit (nonlinear) DAE

$$\begin{aligned} z' &= f[z(t), u(t), t] && \text{(1.8a)} \\ 0 &= g[z(t), u(t), t] && \text{(1.8b)} \end{aligned}$$

- Fully Implicit (nonlinear) DAE

$$F(z', z, t) = 0 \qquad (1.9)$$

The extension of our algorithm to these problems will be left to future research.

The theory for (1.5) and (1.6) is fairly well understood. For (1.5), solvability is expressed in terms of a *matrix pencil.* For square matrices E and F, and complex parameter λ, $\lambda E + F$ is called a matrix pencil. If its determinant is not identically zero as a function of λ, then the pencil $\lambda E + F$ is said to be *regular*. Equation (1.5) is solvable if and only if $\lambda E + F$ is a regular pencil [5]. If (1.5) is solvable we can let $z = Qw$ and premultiply by P so that (1.5) becomes

$$PEQw' + PFQw = Pf(t) = g(t) \qquad (1.10)$$

where P, Q are nonsingular matrices such that

$$PEQ = \begin{bmatrix} I & 0 \\ 0 & N \end{bmatrix}, \quad PFQ = \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix}.$$

N is a nilpotent matrix the index of which is the same as the uniform differentiation index of DAE (1.5). The system is then decoupled, and can be written as

$$\begin{aligned} w_1' + Cw_1 &= g_1(t) && \text{(1.11a)} \\ Nw_2' + w_2 &= g_2(t). && \text{(1.11b)} \end{aligned}$$

Equation (1.11a) is an ODE for which a solution exists for any initial value of w_1 and any continuous forcing function $g_1(t)$. The unique solution to (1.11b) is

$$w_2 = (ND + I)^{-1} g_2(t) = \sum_{i=0}^{\nu-1} (-1)^i N^i g_2^{(i)}(t)$$

where ν is the index, or degree of nilpotency of N, and D is the differentiation operator. Note that the initial values of w_2 are completely determined.

In the linear time-varying case, (1.6), a similar result holds. While the nature of the matrix pencil $\lambda E(t) + F(t)$ is no longer an indicator of solvability, the form of (1.11) is still important in linear time-varying DAEs.

Definition 1.3. *The system (1.6) is in* standard canonical form *if it is in the form*

$$\begin{bmatrix} I & 0 \\ 0 & N(t) \end{bmatrix} z' + \begin{bmatrix} C(t) & 0 \\ 0 & I \end{bmatrix} z = f(t) \tag{1.12}$$

where N is strictly lower (or upper) triangular.

If $E(t)$, $F(t)$ are real analytic, then (1.6) is solvable if and only if, after linear time-varying coordinate changes, it can be written as (1.12). The problem exists in the difficulty of finding those coordinate changes that allow us to write the DAE in standard canonical form.

For the other three cases mentioned above, (1.7)-(1.9), the theory is much newer and also much less understood. For the purpose of this thesis, it is sufficient to note that the concepts presented above serve as a basis for the development of the theory for these cases. It should be noted that while this newer theory is beyond the scope of this thesis, the common starting point serves as a good indicator that the algorithm developed herein for linear time-invariant and linear time-varying DAEs may have applications in the more general case as well.

While there are many more interesting and useful items in the theory of DAEs, these few properties and definitions that we have mentioned will suffice for our purposes. For a complete treatment of DAEs see [5, 7, 8, 9, 10, 13, 14].

1.1.2 Numerical Solutions

While it is not our goal to present an exhaustive overview of the numerical methods that can be used to solve descriptor systems, we briefly mention those methods which will be used in later chapters. The discretization methods we will review are those that are used by the commercial software in which we implement the FDI algorithm developed in this thesis, namely the trapezoidal method, the Compressed Hermite-Simpson method, and the 4-stage implicit Runge-Kutta method. These direct transcription discretizations will be described using the semi-explicit DAE form (1.8). After discretization, several methods exist for solving the resulting finite dimensional problem. Of those methods, only the sparse quadratic program (SQP) will be described here. While the software, Boeing's Sparse Optimal Control Software (SOCS), which will be introduced in a later chapter, can solve DAEs via an analytic transformation, as well as Euler's and linear multistep methods, these schemes will not be used in this thesis, and thus will not be mentioned here. Many of the approaches applied to DAEs are described in detail in [5, 13].

For our discussion of discretization and finite dimensional problem solution, consider a simple optimization problem based on the semi-explicit DAE (1.8)

$$\min_{t_0 \le t \le t_f} J[x(t), u(t), t] \tag{1.13a}$$

subject to

$$\begin{aligned} x' &= f[x(t), u(t), t] && (1.13b) \\ 0 &= g[x(t), u(t), t]. && (1.13c) \end{aligned}$$

Discretization

In general, transcription discretization schemes start by dividing the time interval, $[t_0, t_f]$, into n segments

$$t_0 < t_1 < t_2 < ... < t_n = t_f$$

where the points t_k, $k = 0, ...n$, are referred to as mesh points. Let $x_k = x(t_k)$ be the value of a state variable at a mesh point. Likewise, denote the value of a control at a mesh point as $u_k = u(t_k)$. Let $f_k = f(x_k, u_k, t_k)$ and $g_k = g(x_k, u_k, t_k)$ be the right-hand sides of (1.13b) and (1.13c), respectively. Finally, let $h_k = t_k - t_{k-1}$ be the step size for $k = 1, ..., n$.

Utilizing this notation, the trapezoidal method approximates the state equations (1.13b) and algebraic constraints (1.13c) as

$$\begin{aligned} x_k &= x_{k-1} + \frac{h_k}{2}(f_k + f_{k-1}) && (1.14a) \\ 0 &= g_k. && (1.14b) \end{aligned}$$

In the Compressed Hermite-Simpson scheme we denote the value of the control at the midpoint of a segment as $\overline{u}_k = u(\overline{t}_k)$ where $\overline{t}_k = \frac{1}{2}(t_k + t_{k-1})$, for $k = 1, ..., n$. The discrete approximations for this method are given by

$$\begin{aligned} x_k &= x_{k-1} + \frac{h_k}{6}(f_k + 4\overline{f}_k + f_{k-1}) && (1.15a) \\ 0 &= g_k && (1.15b) \end{aligned}$$

where

$$\overline{f}_k = f(\overline{x}_k, \overline{u}, \overline{t}_k)$$

with

$$\overline{x}_k = \frac{1}{2}(x_{k-1} + x_k) + \frac{h_k}{8}(f_{k-1} - f_k)$$

for $k = 1, ..., n$. The 4-stage implicit Runge-Kutta discretization uses four intermediate, implicit steps

$$\begin{aligned} c_1 &= h_k f(x_{k-1}, u_{k-1}, t_{k-1}) && (1.16a) \\ c_2 &= h_k f(x_{k-1} + \frac{c_1}{2}, \overline{u}_k, \bar{t}_k) && (1.16b) \\ c_3 &= h_k f(x_{k-1} + \frac{c_2}{2}, \overline{u}_k, \bar{t}_k) && (1.16c) \\ c_4 &= h_k f(x_{k-1} + c_3, u_k, t_k). && (1.16d) \end{aligned}$$

The discrete approximations for this method then become

$$\begin{aligned} x_k &= x_{k-1} + \frac{1}{6}(c_1 + 2c_2 + 2c_3 + c_4) && (1.17a) \\ 0 &= g_k && (1.17b) \end{aligned}$$

where $\overline{u}_k$ is defined as before.

These methods have all been proven to converge for index one DAEs, and are thus appropriate for our purpose [4, 5]. In every case, the result of the discretization, when combined with the cost function, J, is a sparse nonlinear programming (NLP) problem. The variables of the problem are the discretized states, x_k, controls, u_k, and time, t_k, for $k = 0, ...n$.

Solving the Finite Dimensional Problem

One way to solve this NLP problem, and the approach used by SOCS, is via a SQP approach [3]. Dropping subscripts for now, let w be the vector of state and control variables, (x, u), and let $F(w, t)$ be the constraint set resulting from the discretization of the DAE. That is, (1.14), (1.15), or (1.17), after shifting everything to the right hand side, becomes

$$0 = F(w, t).$$

Note that F is a function of the state, control, and time variables at all time steps. The SQP algorithm requires an initial guess, w^0, and forms a new iterate by adding a scalar multiple, α, of the search direction, p. That is,

$$w^1 = w^0 + \alpha p.$$

The search direction is found by solving a quadratic programming (QP) subproblem defined at the current point. The QP subproblem is defined as

$$\min J_w^T p + \frac{1}{2} p^T H p$$

subject to

$$0 = Gp$$

where J_w is the gradient vector of the cost function, H is an approximation to the Hessian matrix of the Lagrangian of the NLP ($L = J - \lambda^T F$), and G is the Jacobian matrix of gradients of the constraints F. The step length, α, is computed such that H remains positive definite. The QP subproblem can be solved via either a sparse Schur-Complement method, when appropriate, or a null-space quadratic programming algorithm when G and/or H are dense. Details about the latter can be found in [3].

An algorithm based on the combination of a direct transcription scheme and the SQP approach begins with a discretization and an initial guess. The SQP problem is then solved via the QP subproblem iteration. After each QP subproblem is solved, the current point is updated and the procedure is repeated. The subproblem iteration terminates when a point is reached which satisfies necessary conditions for a local minimum within a given set of tolerances. The solution is then compared to that of the previous discretization iteration, or the initial guess if it is the first iterate. The mesh is refined, the problem re-discretized and the process is repeated until successive iterates agree within an additional given set of tolerances. The QP subproblem demonstrates quadratic convergence under the right conditions [3]. Convergence rates for the direct transcription schemes, when applied to index one descriptor systems, are at least quadratic, and, under the right system coefficient conditions, often are considerably better [5].

1.2 Fault Detection and Isolation

With this basic understanding of the theory of descriptor systems and numerical methods for their treatment, we now turn our attention to the various approaches for treating faults in those systems. We begin with basic control theory, and then turn to feedback, the link between control theory and FDI. Following that is a discussion of the elements of optimal control and H_∞ control pertinent to our approach. We conclude with a discussion of existing research into FDI in descriptor systems and the methods used.

1.2.1 Basic Theory

A descriptor system is one possible result of a system design problem. The problem begins with a task to be accomplished, and the design engineer is usually given goals or objectives that describe the desired performance characteristics of the system along with a set of constraints by which the system is bound. The development of a system which accomplishes the objectives while meeting the constraints is the system design problem.

A particular type of system design problem is the control problem, in which the goal is to generate certain outputs from the system or to maintain the state of the system within certain bounds. For example, an engineer might be asked to design a satellite attitude control system which does not consume too much fuel [1]. The essential elements of such a control problem are

- a mathematical model of the system,
- a desired output,
- a set of admissible controls,
- a performance measure.

Often, as stated above, a descriptor system is the natural product of the system design problem. For the remainder of this thesis, we will restrict most of our study to linear time-invariant systems (1.5). Comments extending our algorithm to linear time-varying systems (1.6) are included in a later chapter.

Consider a system based on the linear time invariant DAE (1.5)

$$\begin{aligned} x' &= Ax + Bu && \text{(1.18a)} \\ y &= Cx && \text{(1.18b)} \end{aligned}$$

where x, y, and u are the state, output, and control vectors, respectively, and the time interval considered is $t \in [t_0, t_f]$. Systems often allow for noise or unknown inputs by adding a term to each equation of (1.18)

$$\begin{aligned} x' &= Ax + Bu + M\mu && \text{(1.19a)} \\ y &= Cx + N\mu && \text{(1.19b)} \end{aligned}$$

where μ is the noise or unknown input, and the matrices M and N are the weight matrices for the state and output noise, respectively.

Central to the study of system (1.18) are the concepts of controllability, observability, and stability [6].

Definition 1.4. *A linear system is said to be* controllable *at t_0 if it is possible to find some input function $u(t)$, defined over $t \in [t_0, t_f]$, which will transfer the initial state $x(t_0)$ to the origin at some finite time $t_1 \in [t_0, t_f]$, $t_1 > t_0$. If this is true for all initial times t_0 and all initial states $x(t_0)$, the system is* completely controllable.

Definition 1.5. *A linear system is said to be* observable *at t_0 if $x(t_0)$ can be determined from the output function $y_{[t_0,t_1]}$ for $t_0 \in [t_0, t_f]$ and $t_0 \leq t_1$, where t_1 is some finite time, $t_1 \in [t_0, t_f]$. If this is true for all t_0 and $x(t_0)$, the system is* completely observable.

Since controllability describes the ability of the control to affect the system state, it involves the matrices A and B. Likewise, since observability describes the ability of the output to characterize the state, it involves the matrices A and C. Simply stated, the nth-order system (1.18) is controllable if and only if $[sI - A \mid B]$ has rank n for all values of s. The same system is observable if and only if $[sI - A^T \mid C^T]$ has rank n for all values of s. Proofs of these characteristics can be found in [6], along with the requirements for controllability and observability in more complex systems.

The concept of stability helps us deal with systems that may not be controllable and/or observable. Stability is closely related to the eigenvalues of the A matrix. Intuitively, a solution to (1.18) is *stable* if we can stay close to the solution by starting close enough to it via the initial condition. A solution is *asymptotically stable* if, by starting close enough, we converge to the solution. A system is *stabilizable* if all unstable modes are controllable, and *detectable* if all unstable modes are observable. Thus the system can be handled effectively provided all uncontrollable and unobservable modes are stable. This situation can often be tolerated in a control system [6]. For the remainder of this thesis, we will assume that we are dealing only with the controllable and/or observable modes of control systems.

1.2.2 Feedback and Observer Design

The bridge between basic control theory and fault detection is the concept of feedback. In a feedback control system, the control, $u(t)$, is modified by information about the system. Sensors measure either the system state, or the output, and then pass that information to the controller, which adjusts the control based on the input from the sensors. One of the fundamental goals of feedback compensator design is to improve the performance of the system through eigenvalue placement. As stated earlier, stability depends on the eigenvalues of the A matrix. By assigning desirable values to eigenvalues, system stability can be enhanced. For the state feedback case, the relation

$$u(t) = Fv(t) - Kx(t) \tag{1.20}$$

is used, where the matrix K is called the feedback gain matrix, and F the feed-forward matrix. Substituting into (1.18), we obtain

$$\begin{aligned} x' &= (A - BK)x + BFv && \text{(1.21a)} \\ y &= Cx. && \text{(1.21b)} \end{aligned}$$

Clearly, the eigenvalues of the $A - BK$ matrix now determine the stability of the system. By careful construction of the feedback gain matrix K, the eigenvalues are assigned the desired values. For the output feedback case, the relation

$$u(t) = Fv(t) - Ky(t) \tag{1.22}$$

is used, where the K and F matrices are as defined above. Substituting this relation into (1.18), we obtain

$$x' = (A - BKC)x + BFv. \tag{1.23}$$

Now the eigenvalues of the $A - BKC$ matrix determine the stability of the system. Unfortunately, due to the presence of the C matrix in this expression, output feedback usually cannot place all of the eigenvalues of the system. This limitation is present when the rank of KC is less than the rank of K, i.e., when C is a long matrix. It should be noted that state feedback may impact the observability of a system, but can have no impact on controllability. Output feedback can impact neither controllability nor observability of a system [6].

Using feedback, the basic tool for many FDI approaches can be constructed: the observer. For most systems the only information about the system state is through the output vector, which often provides only partial information. Thus, output feedback is the only option, and not all system eigenvalues can be placed where desired. To improve system stability in these cases, the most frequently used method is to reconstruct information about the remaining elements of the state vector through development of an observer of the system. Consider the observer

$$\hat{x}' = A\hat{x} + Bu + L(y - C\hat{x}) \tag{1.24}$$

where $\hat{x}$ is the observer estimate for the state vector. Note that y is the output from the real system, (1.21), and $C\hat{x}$ is the observer output. Subtracting (1.18a), and letting $e = \hat{x} - x$ be the observer error, we obtain

$$e' = (A - LC)e.$$

Since L is arbitrary and (A, C) is observable, we can guarantee that observer error goes to zero by selecting L so that $A - LC$ is stable. With this construction, state feedback is possible using the observer estimate for the state vector. Thus all system eigenvalues can be placed where desired, and complete control over system stability is possible. It should be noted that since the complete state vector is reconstructed by the observer, faults which send the system into unpredicted or undesirable states may be detectable by such an observer simply by comparing the observer estimate with those elements of the system state vector which are available. This fault detection can be accomplished without using the observer to affect any feedback compensator for the system.

1.2.3 Optimal Control

Later, when we develop our algorithm, we will work with a control system which acts as the constraints in an optimization problem. This *optimal control* structure is key to the multi-model approach to FDI, which we will discuss in Section 1.2.5. Accordingly, we briefly review optimal control theory. While this area of study is vast, the only topic which we will need for our discussion is the state regulator problem, also called the linear quadratic regulator (LQR) problem. Consider the optimization problem

$$J(x,u) = \min \ \frac{1}{2}x(t_f)^T S_f x(t_f) + \frac{1}{2}\int_{t_0}^{t_f} x^T Q x + u^T R u \, dt \tag{1.25a}$$

subject to

$$x' = Ax + Bu \tag{1.25b}$$

as well as some initial conditions at the beginning of the interval, where S_f, Q, and R are the weight matrices for the terminal cost, the trajectory, and the control. It is assumed that Q is positive semi-definite and R is positive definite. This is one form of the LQR problem and it is important for three reasons. First, the theory is elegant and robust. Results are easy to understand and implement in numerical algorithms. Second, it has strong geometry. $J(x,u)$ is actually an inner product norm with useful properties. Finally, there are strong physical correlations to this type of problem. Energy is a quadratic form, as is power.

As with any optimization problem, the LQR problem possesses necessary conditions for a minimum. For the problem (1.25), we construct the Hamiltonian

$$H(x,\lambda,u) = \frac{1}{2}(x^T Q x + u^T R u) + \lambda^T(Ax + Bu) \tag{1.26}$$

where λ is the Lagrange multiplier for the constraints. The Euler equations, which must be satisfied by any extremum of the problem, are

$$
\begin{aligned}
H_\lambda^T &= x' && \text{(1.27a)}\\
-H_x^T &= \lambda' && \text{(1.27b)}\\
H_u^T &= 0. && \text{(1.27c)}
\end{aligned}
$$

When applied to (1.26), we obtain

$$
\begin{aligned}
\lambda' &= -Qx - A^T\lambda && \text{(1.28a)}\\
x' &= Ax + Bu && \text{(1.28b)}\\
0 &= Ru + B^T\lambda. && \text{(1.28c)}
\end{aligned}
$$

Using (1.28c) and our assumption that $R > 0$, we can eliminate u from (1.28) to obtain a set of differential equations in x and λ

$$
\begin{aligned}
x' &= Ax - BR^{-1}B^T\lambda && \text{(1.29a)}\\
\lambda' &= -Qx - A^T\lambda. && \text{(1.29b)}
\end{aligned}
$$

While this form will be useful in our algorithm, it is possible to take an additional step and eliminate λ, resulting in a matrix Riccati differential equation for the optimal control feedback gain matrix. The derivation of the Riccati equation will be detailed when we develop our algorithm in the next chapter. It should be noted that our assumptions on the Q and R matrix, while not restrictive in an applicability sense, guarantee that the extremum which satisfies the necessary conditions represents at least a local minimum of the cost $J(x, u)$. In fact, Q is often positive definite, and in that case, the conditions for an extremum are necessary and sufficient. Detailed discussions of this and other topics in optimal control can be found in [1, 6, 38].

1.2.4 H_∞ Control

H_∞ control in the time domain is similar to optimal control. It takes advantage of the linear quadratic (LQ) form in addressing significant uncertainties in the energies of system noises. For bounded energy noise inputs, where little or no other knowledge is available about the signal, the LQR formulation is an elegant worst-case approach. The model generally takes the form of (1.19), and all functions are assumed to exist in the space of square integrable functions, denoted L^2. While H_∞ performance criteria vary, they all share the structure of the optimal control cost function, that is, they are all in LQ form.

In this setting, filtering, smoothing, and compensator design are efficiently accomplished. Nagpal and Khargonekar [27] apply a filtering and smoothing method using an H_∞ performance criterion on both the finite and half-infinite intervals to accomplish state estimation (filtering) and output smoothing. Tadmor [36] attempts to find, in LQ game-theoretic terms, the compensator which provides the best control in response to the worst disturbance. Matrix Riccati equations provide solutions in each case.

While the structure of our problem is very similar to the H_∞ problem, several key differences will become apparent. First, we will solve a different problem. While Tadmor [36] designs a worst-case compensator, and Nagpal and Khargonekar [27] solve for the optimal filter and smoother in the face of various initial conditions, we will solve for the optimal fault detection signal. In addition, while both [27] and [36] work in single model systems, we will work in a multi-model system. Finally, while the noise present in our system is also L^2, it is not the same kind of signal as is commonly assumed in H_∞ control. The impact of these differences will become significant as our problem is defined and our algorithm developed.

1.2.5 Prior Research

In addition to the two approaches mentioned above, fault detection and isolation in linear control systems has been attempted from many angles. To begin, we note that there exist two basic types of approaches to FDI: passive and active. In the passive approach, only monitoring of system performance is allowed. No interaction with the system occurs, either for material or security reasons. The system states (or outputs) are measured and compared to "normal" system behavior, generating a residual. The residual is computed such that it is equal or close to zero when no faults are present, and much different from zero when a fault occurs. The vast majority of research in FDI using the passive approach applies observers to generate residuals.

Passive Methods

Nuninger et al. [32] use analytic redundancy in order to detect sensor and actuator failures or process disturbances. Analytic redundancy attempts to generate a residual that might contain information about the faults. Two methods for generating the residual are examined. First, direct residual generation is based on the parity space approach, using the input-output transfer function (the parity equation). Second, indirect residual generation is based on output estimation, using an observer to do state estimation first. The authors apply the first method to known input systems and the second method to both known and unknown input systems. A drawback of this approach is that some faults may have no influence on the residuals generated by either method, so perfect FDI is not attained. Chen and Speyer [15] also use analytic redundancy, generating a residual via an observer that reconstructs the system state vector. Their model has the target fault direction explicitly in the detection filter gain calculations, allowing for enhanced sensitivity of the filter to the target fault.

Koenig et al. [24] present a comparative study of several design methods for unknown input observers (UIO) used for FDI and Correction. Their goal is to design an integrated approach which can detect, isolate, and correct a large variety of faults for a desired system with real-time computation constraints. Methods compared are: failure isolation by using banks of observers (robust to some faults, but sensitive to others, in combination so that all faults are detectable), failure isolation by observer pole placement (to create an unknown input fault detection observer), and failure correction via general structured UIO (design of full order observer to estimate states as well as unknown inputs). Chung and Speyer [17] develop a game theoretic detection filter, which is similar to the UIO, that attenuates disturbances, bounding all signals except the fault to be detected, embedding the exogenous signals into an unobservable, invariant subspace. The subspace structure is used to reduce the order of the limiting filter by factoring the invariant subspace out of the state space, resulting in a lower order filter sensitive only to the fault to be detected. The filter is applied to the flight control characteristics of the F-16XL and a simple rocket.

The parity relation approach to residual genertion is applied by Youssouf and Kinnaert [40]. The method is based on the inverse of the map from both unknown inputs and faults to the observable signals (measured inputs and outputs), using a variable change in the frequency domain. Tools available for nonsingular systems can be used on the resulting map. The authors contend that FDI for singular systems depended previously on state estimation, which put unnecessary requirements on the plant, as there is no need to reconstruct the entire state vector to do residual generation. Where Youssouf and Kinnaert [40] apply their method to continuous time systems, Sauter et al. [35] do the same for the discrete case in mechanical systems, though the theory and algorithm are completely different. They do state equation decoupling before residual generation, which is contrary to customary methods. The decoupling involves separating out the unknown input from the system state instead of from the residual.

Chowdhury and Aravena [16] go in a slightly different direction. They apply a modular methodology to the area of fast fault detection and classification in dynamical electrical power systems. Module I is the generation of fault indicators in one of two ways:

1. model-based, in which a residual is generated using either an accurate mathematical or I/O model of the nominal system, or an I/O model is built on-line, which is very difficult,

2. model-free, in which detectable variables are measured and enhanced if necessary by signal-processing techniques.

The authors present a model-free orthogonal decomposition based on multirate filter banks to produce a fault indicator. Module II is the measuring and testing of fault indicators via either statistical test or feedforward neural-network testing. The authors explore the neural network aspect. Fault classification occurs in module III, another neural network, the operation of which depends on the existence of a system model. The emphasis is on model-free methods, those lesser explored and lesser restrictive cases where models are not available, non-linearities prevent model derivation, or too many uncertainties exist in the system. These cases appear to hold the most promise for neural-network applications in fault detection.

Hybrid Passive-Active Methods

Some research has been done using a hybrid of the passive/active approaches. The passive approach is used to detect faults, then an active approach is used to correct or compensate for faults through feedback. Zhang and Jiang [43] investigate the application of integrated fault detection, diagnosis, and reconfigurable control to discrete-time stochastic vertical take-off and landing aircraft systems. A bank of two-stage adaptive Kalman filters is used for FDI, and statistical decisions are made for fault detection, diagnosis and activation of controller reconfiguration. In a related paper [42], the same authors apply an interacting multiple-model based approach to the same type of control problem. A finite-state Markov chain is linked to the same Kalman filter bank for fault diagnosis. The decision from this process is used to activate system reconfiguration via eigenstructure assignment.

Active Methods

The drawback inherent in the passive approach is that faults can be masked by the application of the control. Thus it is possible that a fault could go undiscovered until it is too late to correct it. In direct contrast, the active approach interacts with the system on a periodic basis, or at critical times, to detect faults, thus eliminating the possibility of the presence of undetected faults. The approach uses various types of interaction with the system to detect faults. A test signal, which is constructed in such a way that faults are highlighted, is fed into the system. Observation and/or manipulation of the resulting output is used to make a decision about system faults. Observers designed to aid in feedback, as well as various other types of feedback compensators, are examples of the active approach.

Bennett et al. [2] apply speed dependent feedback (a stable time-varying linear observer) to detect intermittent, short duration faults in bilinear dynamical systems. The AC drive system for an electric train is considered. These systems experience abrupt disconnections which introduce severe transient errors and which are hard to detect due to their short durations. The parity equation approach is not preferred in this case due to the intermittent nature of the faults. By combining the observer and Kirchoff's law, a bank of observers is constructed to detect and correct disconnections.

This case is an example of the application of a test signal as part of the feedback to control the system and correct faults.

The multi-model approach is well-suited to the case where it is desirable to apply a test signal independent of feedback control. The approach relies on the presence of the system model

$$x_i' = A_i x_i + B_i v + M_i \mu_i \quad (1.30a)$$

$$y = C_i x_i + N_i \mu_i \quad (1.30b)$$

for $i = 0, ..., m$, where m is the number of faults expected from the system, and v is the test signal. A different system model exists, with known parameters, for each possible fault. It is assumed that any feedback control has been absorbed into the A_i matrix, thus eliminating the control u from the differential equation. The difficulties in this approach lie in determining from which model an output y derives, as well as the computation of system parameters for each fault model. Nikoukhah [28] presents the use of a test signal for active FDI in discrete-time linear systems subject to inequality-bounded perturbations. Detectability is required, but when present, guaranteed FDI is attained. The discrete time case lends itself to recursive algorithms, and so recursion is used extensively by the author to develop the test signal. After constructing a test signal that separates outputs into disjoint convex sets, the author uses the separating hyperplane approach to determine which set a certain output falls into, and thus whether a fault has occurred. Linear programming is used to construct the separating hyperplane. Nikoukhah et al. [29] has the same goal as [28], but goes about it completely differently. Among the differences, fault isolation is accomplished by a ratio test, and optimal control theory is applied. This paper is the inspiration for our current research, and thus uses some of the same techniques we use, but applies them to discrete time control problems. We apply our methodology to continuous time control problems, which present key theoretic and algorithmic differences. In addition, both [28] and [29] consider only two model systems, whereas our approach can handle problems with three or more models.

The multi-model approach is also useful with Kalman filtering. Keller et al. [21] presents the multi-model approach for fault detection in stochastic systems with unknown inputs. The method uses the two-stage Kalman filter with unknown inputs and constant biases, the first stage of which is bias-free (for fault detection) and the second stage is a bias filter (for fault isolation). The optimum state estimate is expressed as the output of the bias-free filter corrected with the output of the bias filter. Different fault types are detected using a bank of such filters. The two stages of the filter reduce computational time associated with the presence of multiple faults.

1.2.6 Conclusion

As mentioned in the introduction to this chapter, the combination of the multi-model approach and the bounded energy noise model seems to be under-explored. The common thread running through most of the applications mentioned in the last section is the modeling of noise. [16, 21, 22, 23, 33, 37, 41, 42, 43] model noise as some type of random variable. Many use filtering or statistical tests to make the fault decision, and thus do not model noise at all. Only [17, 27, 28, 29, 36] model noise as bounded energy signals. As we shall see, the bounded energy noise model is very suited to the multi-model approach, and the combination as developed in this thesis provides a powerful tool for fault detection and isolation in descriptor systems.

1.3 Outline of Thesis

In the next chapter, we present the theory and algorithm for the fault detection phase of the problem, along with extensions of the method to variations of the basic problem. Following that, Chapter 3 is the development of the algorithm for the model identification phase. In Chapter 4, we state the full algorithm, then present and analyze several examples. Lastly, Chapter 5 is the conclusion and outline of future research possibilities in this area. Software codes for the algorithm are in Appendix A.

1.4 Contributions of Thesis

The research in this thesis will appear, or has already appeared in the following publications:

- S. L. Campbell, K. Horton, R. Nikoukhah, and F. Delebecque, *Rapid Model Selection and the Separability Index*, in Proc. 4th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS 2000), Budapest, Hungary, June 2000, pp. 1187-1192.
- R. Nikoukhah, F. Delebecque, S. L. Campbell, and K. Horton, *Multimodel Identification and the Separability Index*, in Proc. 14th International Symposium of the Mathematical Theory of Networks and Systems 2000, Perpignan, France, June 2000, CDROM.
- R. Nikoukhah, S. L. Campbell, Kirk Horton, and F. Delebecque, *Auxiliary signal design for robust multi-model identification*, IEEE Transactions on Automatic Control, accepted subject to final revision.
- S. L. Campbell, Kirk Horton, R. Nikoukhah, and F. Delebecque, *Auxiliary signal design for rapid multi-model identification using optimization*, submitted to Automatica.

Chapter 2

Fault Detection via the Detection Signal

2.1 The Problem - Finding the Minimum Energy Detection Signal

As introduced in the previous chapter, our goal is to attain near-perfect fault detection and model identification in linear descriptor systems using the multi-model approach. This approach allows the treatment of the problem in two steps. In this chapter, our focus will be on the fault detection step of the problem, while the next chapter will tackle the model identification step.

Multi-model fault detection and model identification means that we have two or more possible models for a system, and we decide which one corresponds to the system based on measurements of the inputs and outputs of the system over a finite test period, $[0, t_f]$. While other possible test periods exist, we will restrict our discussion to the finite interval.

In order to exclude all but one model based on input-output measurements, the input signal must have special properties. A signal with such properties is called a *proper detection signal.* For the remainder of the present discussion we will assume that two possible models exist for the system: the nominal, or unfailed model, and the fault model. This assumption is not restrictive in any way, and later we will describe how the algorithm can be extended to include the case in which more than one fault model is present.

2.1.1 Problem Setup

The true model of the system is one of two models

$$x_i' = A_i x_i + B_i v + M_i \mu_i \tag{2.1a}$$

$$y = C_i x_i + N_i \mu_i \tag{2.1b}$$

for $i = 0$ and 1, and for $t \geq 0$, where x_i, y, v, and μ_i are the system states, output, detection signal, and noise, respectively. The matrices A_i, B_i, C_i, M_i, and N_i are matrices of appropriate dimensions. We assume that v and μ_i are in $L^2[0, t_f] = L^2$, forcing x_i and y to be in L^2 as well. While we assume full row rank of the M_i and N_i, and controllability/observability of the system for computational reasons, there is no assumption that the dimensions of the state or noise vectors of the two models are the same. We also assume no *a priori* information about the system before $t = 0$, and in particular no information about $x_i(0)$. Thus, unlike some existing theory, in particular [30], we have no weights on $x_i(0)$. (We will discuss the impact of information about initial conditions and the subsequent presence of weight matrices on $x_i(0)$ later in this chapter.) In addition, we assume that any feedback control has been absorbed into the A_i matrices as described in Chapter 1, or else is nulled at $t = 0$ for the duration of the test period. Thus, the only common elements of the two models are the output, y, and the detection signal, v. Note that (2.1) is a linear descriptor system since the output y is known.

Consider the detection signal v and let $\mathcal{A}^0(v)$ be the set of possible outputs associated with this input if Model 0, the nominal model, is the correct model. Likewise, let $\mathcal{A}^1(v)$ be the set of outputs if Model 1, the fault model, is the correct model. Then perfect model identification based on output measurement implies that

$$\mathcal{A}^0(v) \cap \mathcal{A}^1(v) = \emptyset. \tag{2.2}$$

This is achievable thanks to the bounded energy noise model. This noise model can be expressed as

$$\mathcal{S}_i(\mu_i) \equiv ||\mu_i||^2 = \int_0^{t_f} |\mu_i(t)|^2 \, dt < 1, \quad i = 0, 1 \tag{2.3}$$

where $|\cdot|$ is the (pointwise) Euclidean norm, and thus $||\cdot||$ is the L^2 norm. In practice one has bounds $||\mu_i||^2 < K$. It is always possible to rescale the M_i, N_i to get $K = 1$, so we assume without loss of generality that $K = 1$.

This expression for the noise allows us to distinguish between the two basic types of detection signals.

DEFINITION 2.1. *The detection signal v is* not proper *if there exist x_0, x_1, μ_0, μ_1, and y satisfying (2.1) and (2.3). The detection signal v is called* proper *otherwise.*

Thus we say that the L^2 vector function v is a proper detection signal if its application implies that we are always able to distinguish the two candidate models based on observation y. That is, condition (2.2) is satisfied [30]. Note that $v = 0$ is not proper since the zero solution is always in the intersection of (2.2). In addition, if v is proper then cv is also proper for $c \geq 1$, but if v is not proper then there exists an $\epsilon > 0$ such that cv is also not proper for $0 \leq c \leq 1 + \epsilon$. These facts will be useful when we develop the optimization problem later in the chapter.

The conditions for the existence of proper detection signals are quite weak. For their characterization, let

$$\mathcal{L}_i(f) = \int_0^t e^{A_i(t-s)} f(s) ds \tag{2.4}$$

be the solution of $z' = A_i z + f$, $z(0) = 0$. Then the solutions to (2.1) are

$$x_i = \mathcal{L}_i(B_i v) + \mathcal{L}_i(M_i \mu_i) + e^{A_i t}\xi_i \tag{2.5a}$$

$$y = C_i \mathcal{L}_i(B_i v) + C_i \mathcal{L}_i(M_i \mu_i) + C_i e^{A_i t}\xi_i + N_i \mu_i \tag{2.5b}$$

for $i = 0, 1$, where ξ_i is the free initial condition for x_i. Thus the output set for each model is the sum of three terms

- $y_i = C_i \mathcal{L}_i(B_i v)$ which is a vector depending linearly on the detection signal, v,
- $\{(C_i \mathcal{L}_i M_i + N_i)\mu_i : \|\mu_i\| < 1\}$ which is an open convex set,
- $\{C_i e^{A_i t}\xi_i : \xi_i \in \Re^n \text{ (or } \mathcal{C}^n)\}$ which is a finite dimensional subspace of L^2.

Because of these facts, and noting that y_0 and y_1 are respectively the outputs of Model 0 and Model 1 corresponding to zero noise and zero initial state, we see that the output sets $\mathcal{A}^0(v)$ and $\mathcal{A}^1(v)$ are translates by y_0 and y_1 of bounded open sets. Since y_0 and y_1 depend linearly on v, either $y_0 = y_1$ for all v, or $y_0 - y_1$ can be made arbitrarily large with proper choice of v. So proper detection signals exist provided the linear mapping of v to y_0 is distinct from the linear mapping of v to y_1 [30]. In the time invariant case, this is equivalent to

$$C_0(sI - A_0)^{-1}B_0 - C_1(sI - A_1)^{-1}B_1 \neq 0 \tag{2.6}$$

for some s.

The amount of energy required for a detection signal to be proper determines the separability of the output sets $\mathcal{A}^0(v)$ and $\mathcal{A}^1(v)$.

Definition 2.2. *Let V denote the set of proper detection signals v. Then,*

$$\gamma^* = \left(\inf_{v \in V} \|v\|^2\right)^{-\frac{1}{2}} \tag{2.7}$$

is called the separability index *associated with (2.1).*

Thus, $(\gamma^*)^{-2}$ is a lower bound on the energy of proper detection signals. Also, the inverse relationship between the separability index and the proper detection signal energy indicates that systems with lower energy proper detection signals have a higher separability index. The separability index is zero if there are no proper detection signals. Later, the algorithm we develop will compute γ^* as the objective function of an embedded optimal control problem. In Section 2.4.5 we describe an existing algorithm that computes γ^* [30]. Our approach has the advantage of being able to address several problems that the algorithm in [30] cannot handle.

2.1.2 Formulation as an Optimal Control Problem

Before we describe the algorithm, however, the problem of finding the minimum energy proper detection signal must be formulated as an optimal control problem. First, note that for the detection signal v to be not proper, (2.1) must hold as well as (2.3). We can rewrite (2.3) as

$$\max\left\{\int_0^{t_f} |\mu_0(t)|^2\, dt, \int_0^{t_f} |\mu_1(t)|^2\, dt\right\} < 1. \tag{2.8}$$

This expression can also be written as

$$\max_{0\leq\beta\leq 1}\int_0^{t_f} \beta|\mu_0(t)|^2 + (1-\beta)|\mu_1(t)|^2\, dt < 1. \tag{2.9}$$

Thus we obtain a useful characterization of not proper detection signals [30]

Lemma 2.1. *The detection signal v is not proper if and only if*

$$\inf\max_{0\leq\beta\leq 1}\int_0^{t_f} \beta|\mu_0(t)|^2 + (1-\beta)|\mu_1(t)|^2\, dt < 1 \tag{2.10}$$

where the infimum is taken over (x_i, μ_i, y) in L^2, subject to (2.1), $i = 0, 1$.

This characterization is useful because the algorithm we develop will compute the minimum energy proper detection signal by finding the detection signal of smallest norm that does not satisfy (2.10).

The next step in formulating the computation of the separability index as an optimal control problem involves dimension reduction. By assumption the N_i are both full row rank. Thus, we can perform a constant orthogonal change of coordinates on the N_i (via a QR decomposition on N_i^T). As a result we obtain

$$N_i = [\overline{N}_i \;\; 0] \tag{2.11}$$

where $\overline{N}_i$ is invertible, and

$$M_i = [\overline{M}_i \;\; \widetilde{M}_i]. \tag{2.12}$$

Let $\mu_i = \begin{pmatrix} \overline{\mu}_i \\ \widetilde{\mu}_i \end{pmatrix}$ with the same decomposition as N_i, and subtract (2.1b) for $i = 1$ from (2.1b) for $i = 0$. Equation (2.1b) becomes

$$0 = C_0 x_0 - C_1 x_1 + \overline{N}_0 \overline{\mu}_0 - \overline{N}_1 \overline{\mu}_1. \tag{2.13}$$

Now we can solve for either $\overline{\mu}_i$ and use the resulting expression to eliminate (2.13) by substituting it into (2.1a). Solving for $\overline{\mu}_0$, we obtain

$$\begin{pmatrix} x_0' \\ x_1' \end{pmatrix} = \begin{bmatrix} A_0 - \overline{M}_0 \overline{N}_0^{-1} C_0 & \overline{M}_0 \overline{N}_0^{-1} C_1 \\ 0 & A_1 \end{bmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} +$$
$$\begin{bmatrix} \widetilde{M}_0 & \overline{M}_0 \overline{N}_0^{-1} \overline{N}_1 & 0 \\ 0 & \overline{M}_1 & \widetilde{M}_1 \end{bmatrix} \begin{pmatrix} \widetilde{\mu}_0 \\ \overline{\mu}_1 \\ \widetilde{\mu}_1 \end{pmatrix} + \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} v. \tag{2.14}$$

With the obvious correspondences, the reduced system, no longer a descriptor system, can be written as

$$x' = Ax + Bv + M\mu. \tag{2.15}$$

Note that we do not require A to be stable. An unstable A is allowable because it includes the case in which the original system is stable, the fault model is unstable, and we desire to detect the fault in a short test period to prevent the instability of the fault from creating problems for the system.

The characterization of not proper, (2.10), for the reduced system becomes

$$\inf \max_{0\leq\beta\leq 1} P(x,\mu,\beta) \ < 1 \tag{2.16}$$

where

$$P(x,\mu,\beta) = \int_0^{t_f} \beta(|-\overline{N}_0^{-1}C_0x_0 + \overline{N}_0^{-1}C_1x_1 + \overline{N}_0^{-1}\overline{N}_1\overline{\mu}_1|^2 + |\widetilde{\mu}_0|^2) + (1-\beta)(|\overline{\mu}_1|^2 + |\widetilde{\mu}_1|^2)\ dt \tag{2.17}$$

and the infimum is now taken over (x,μ) in L^2, subject to (2.15).

The third step in the transformation to an optimal control problem involves using the definition of the Euclidean norm to expand the integrand. After doing so and combining like terms, we can rewrite (2.17) as

$$P(x,\mu,\beta) = \frac{1}{2}\int_0^{t_f} x^TQx + x^TH\mu + \mu^TR\mu\ dt \tag{2.18}$$

where

$$Q = 2\beta\begin{bmatrix} C_0^T\overline{N}_0^{-T}\overline{N}_0^{-1}C_0 & -C_0^T\overline{N}_0^{-T}\overline{N}_0^{-1}C_1 \\ -C_1^T\overline{N}_0^{-T}\overline{N}_0^{-1}C_0 & C_1^T\overline{N}_0^{-T}\overline{N}_0^{-1}C_1 \end{bmatrix} \tag{2.19}$$

$$H = 4\beta\begin{bmatrix} 0 & -C_0^T\overline{N}_0^{-T}\overline{N}_0^{-1}\overline{N}_1 & 0 \\ 0 & C_1^T\overline{N}_0^{-T}\overline{N}_0^{-1}\overline{N}_1 & 0 \end{bmatrix} \tag{2.20}$$

$$R = 2\begin{bmatrix} \beta I & 0 & 0 \\ 0 & (1-\beta)I + \beta\overline{N}_1^T\overline{N}_0^{-T}\overline{N}_0^{-1}\overline{N}_1 & 0 \\ 0 & 0 & (1-\beta)I \end{bmatrix}. \tag{2.21}$$

Note that Q is symmetric, positive semi-definite and R is symmetric, positive definite.

Finally, letting S_v be the set of L^2 functions (x, μ) satisfying the constraints (2.15), and defining

$$J_v(\beta) = \inf_{(x,\mu)\in S_v} P(x, \mu, \beta) \tag{2.22}$$

we call on a useful result [30].

THEOREM 2.1. *The function P has at least one saddle point (x^s, μ^s, β^s) on $S_v \times [0, 1]$ and*

$$\inf_{(x,\mu)\in S_v} \max_{0\leq\beta\leq 1} P(x, \mu, \beta) = \min_{(x,\mu)\in S_v} \max_{0\leq\beta\leq 1} P(x, \mu, \beta) =$$

$$\max_{0\leq\beta\leq 1} \min_{(x,\mu)\in S_v} P(x, \mu, \beta) = P(x^s, \mu^s, \beta^s). \tag{2.23}$$

Proof (from [30]) Let $(\bar{x}^\beta, \bar{\mu}^\beta)$ be the solution of problem (2.22). Then $\mathcal{S}_i(\bar{\mu}_i^\beta)$, $i = 0, 1$, depend continuously on $0 < \beta < 1$. Moreover, since

$$\mathcal{S}_0(\bar{\mu}_0^\beta) = 0, \quad \text{if } \beta = 1, \tag{2.24}$$

$\mathcal{S}_0(\bar{\mu}_0^\beta)$ is continuous for $\beta \in (0, 1]$, and since

$$\mathcal{S}_1(\bar{\mu}_1^\beta) = 0, \quad \text{if } \beta = 0, \tag{2.25}$$

$\mathcal{S}_1(\bar{\mu}_1^\beta)$ is continuous for $\beta \in [0, 1)$. Suppose

$$\lim_{\beta\to 1} \mathcal{S}_1(\bar{\mu}_1^\beta) > 0. \tag{2.26}$$

Then for some $0 < \beta^s < 1$, we must have

$$\mathcal{S}_0(\bar{\mu}_0^{\beta^s}) = \mathcal{S}_1(\bar{\mu}_1^{\beta^s}). \tag{2.27}$$

Let $(x^s, \mu^s) = (\bar{x}^{\beta^s}, \bar{\mu}^{\beta^s})$. Then

$$P(x^s, \mu^s, \beta) \leq \mathcal{S}_1(\mu_1^s), \quad \forall \beta \in [0, 1], \tag{2.28}$$

(holding at equality because β cancels out) and

$$P(x, \mu, \beta^s) \geq \mathcal{S}_1(\mu_1^s), \;\; \forall(x, \mu) \in S_v \tag{2.29}$$

because (x^s, μ^s) is the optimal solution of (2.22) for $\beta = \beta^s$. This implies that (x^s, μ^s, β^s) is a saddle point and the rest follows. Now suppose that (2.26) does not hold so that

$$\lim_{\beta \to 1} \mathcal{S}_1(\bar{\mu}_1^\beta) = 0. \tag{2.30}$$

In that case $\mathcal{S}_0$ and $\mathcal{S}_1$ can be made arbitrarily small simultaneously. This implies that $J_v(\beta) = 0$ for all β which means that there exists (x^s, μ^s) such that (2.27) holds with equality to zero. Then, clearly (2.28) holds because both sides of the inequality are zero. In addition, (2.29) holds for all $\beta^s \in [0, 1]$ because the right hand side of the inequality is zero and the left hand side cannot be negative. This implies that (x^s, μ^s, β^s) is a saddle point and the rest follows. □

Note that the above proof in [30] is done with knowledge of, and weight matrices on the initial state, $x_i(0)$. In that case, the bounded energy noise model becomes

$$\mathcal{S}_i(x_i(0), \mu_i) \equiv x_i(0)^T F_{i,0} x_i(0) + \int_0^{t_f} |\mu_i(t)|^2 \, dt < 1, \quad i = 0, 1. \tag{2.31}$$

Since each $\mathcal{S}_i$ is the sum of positive semi-definite terms, letting one term go to zero does not alter the proof.

This result allows us to interchange the order of the inf and the max in (2.16), and replace inf with min. Thus

$$J_v(\beta) = \min_{(x,\mu) \in S_v} P(x, \mu, \beta) \tag{2.32}$$

and, the characterization of not proper becomes

$$\max_{0 \leq \beta \leq 1} J_v(\beta) < 1. \tag{2.33}$$

Expanding this result to its fully explicit form, we see that a detection signal v is not proper if and only if

$$\max_{0\leq\beta\leq 1} \min \frac{1}{2}\int_0^{t_f} x^TQx + x^TH\mu + \mu^TR\mu \, dt < 1 \tag{2.34}$$

where the min is subject to

$$x' = Ax + Bv + M\mu. \tag{2.35}$$

The inner minimization, the $J_v(\beta)$ problem, is a standard LQR optimal control problem with an added cross term in the objective function and the forcing function Bv in the constraint. $J_v(\beta)$ is called the *auxiliary cost function* for the problem.

The auxiliary cost function exhibits several useful qualities [12].

LEMMA 2.2. *For all $v \in L^2$, for $0 \leq \beta \leq 1$, $J_v(\beta)$ is defined and has the following properties:*

1. *It is zero for $\beta = 0$ and $\beta = 1$,*
2. *It is quadratic in v, i.e., for all scalar c, $J_{cv}(\beta) = |c|^2 J_v(\beta)$,*
3. *It is a continuous function of β,*
4. *If v is not proper, then $J_v(\beta) < 1$ for all $0 \leq \beta \leq 1$. Equivalently, $J_v(\beta) \geq 1$ for some β implies v is proper,*
5. *It is a strictly concave function of β if the set of proper detection signals is not empty, otherwise it is identically zero.*

The proof is straightforward and relies on continuity and linearity. It can be found in [12]. With this result, we can state the original problem of finding a minimum energy proper detection signal v as

$$\min \|v\| \;\; \text{subject to} \;\; \max_{0<\beta<1} J_v(\beta) \geq 1. \tag{2.36}$$

Note that the cases $\beta = 0$ and $\beta = 1$ are excluded because $J_v(0) = J_v(1) = 0$, and Lemma 2.2 demonstrates continuity of $J_v(\beta)$ at these points.

Using the fact that $J_v(\beta)$ is quadratic in v, we arrive at the following fundamental result

THEOREM 2.2. *Let*

$$J^*(\beta) = \sup_{v \neq 0} \frac{J_v(\beta)}{\int_0^{t_f} |v|^2 dt} = \sup_{\|v\|=1} J_v(\beta). \tag{2.37}$$

Then

$$(\gamma^*)^2 = \max_{0<\beta<1} J^*(\beta) \tag{2.38}$$

where γ^* is the separability index defined previously.

This theorem, while similar to results in [29] and [30], has added technical difficulties due to the presence of the infinite dimensional space of the independent variable and the unbounded finite dimensional subspace of the output sets. Despite these differences, the proof is an extension of that in [29]. However, it is somewhat technical and requires functional analysis and convergence theory for sequences. See [12] for the complete proof.

Note that the ease of separating the nominal and fault models of a system is proportional to the size of γ^*. When $\gamma^* = 0$, the models are indistinguishable regardless of the detection signal.

As a final result before defining the optimization problems we will address, we state a useful corollary to Lemma 2.2.

COROLLARY 2.1. *A detection signal v is proper if and only if $J_v(\beta) \geq 1$ for some $0 < \beta < 1$.*

Proof Lemma 2.2, part 4, shows that v is proper if $J_v(\beta) \geq 1$ for some β. To show the converse suppose that $J_v(\beta) < 1$ for all β. For each β, let $\{\mu_0(\beta), \mu_1(\beta)\}$ be where $J_v(\beta)$ attains its minimum. Clearly, the values producing a minimum at each β endpoint are $\mu_1 = 0$ for $\beta = 0$, and $\mu_0 = 0$ for $\beta = 1$. Thus there will be a value $\overline{\beta}$ where $\|\mu_0(\overline{\beta})\| = \|\mu_1(\overline{\beta})\|$. But then $\|\mu_i(\overline{\beta})\| < 1$, which shows that v is not proper. □

2.1.3 Problem Statement

We can now state the two versions of the problem solved by the first half of the algorithm. Version One, from (2.37-2.38) is:

$$(\gamma^*)^2 = \max_{\substack{\|v\| = 1 \\ 0 < \beta < 1}} J_v(\beta). \tag{2.39}$$

Version Two, from (2.7) and (2.36) is:

$$(\gamma^*)^{-2} = \min_{\substack{J_v(\beta) \geq 1 \\ 0 < \beta < 1}} \int_0^{t_f} |v|^2 dt. \tag{2.40}$$

These problems will be solved by first calculating the necessary conditions for a minimum of the inner problem which defines $J_v(\beta)$, then numerically solving the outer problem using the previously computed necessary conditions as constraints.

2.2 Necessary Conditions

As with many types of optimization problems, the $J_v(\beta)$ problem possesses conditions that any extrema must satisfy in order to be an optimal solution. In Chapter 1 we introduced the necessary conditions for an optimal solution to the standard LQR problem. The $J_v(\beta)$ problem, while similar, is not the same problem as that discussed in Chapter 1 because of the presence of the cross term in the integral, so in this section we develop the necessary conditions for the $J_v(\beta)$ problem explicitly.

2.2.1 Computing the Necessary Conditions

Recall from (2.32) that

$$J_v(\beta) = \min \frac{1}{2} \int_0^{t_f} x^T Q x + x^T H \mu + \mu^T R \mu \, dt \tag{2.41a}$$

subject to

$$x' = Ax + Bv + M\mu. \tag{2.41b}$$

The Hamiltonian for system (2.41) is

$$\widehat{H} = \frac{1}{2} x^T Q x + \frac{1}{2} x^T H \mu + \frac{1}{2} \mu^T R \mu + \lambda^T (Ax + Bv + M\mu). \tag{2.42}$$

As described in Chapter 1, the Euler equations for an extremum are

$$\begin{aligned} \widehat{H}_\lambda^T &= x' && (2.43a) \\ -\widehat{H}_x^T &= \lambda' && (2.43b) \\ \widehat{H}_\mu^T &= 0. && (2.43c) \end{aligned}$$

These conditions applied to (2.42) give

$$\begin{aligned} x' &= Ax + Bv + M\mu && (2.44a) \\ \lambda' &= -Qx - \frac{1}{2} H\mu - A^T \lambda && (2.44b) \\ 0 &= R\mu + \frac{1}{2} H^T x + M^T \lambda. && (2.44c) \end{aligned}$$

which is an index one DAE in (x, λ, μ) since $R > 0$.

While our algorithm will use (2.44) in their current form, it will be beneficial to define these conditions in terms of a matrix Riccati equation as well. Riccati equations stabilize relatively quickly, and are thus useful in theoretical developments involving long time intervals. The derivation also lends insight into boundary conditions for (2.44). In addition, Riccati equations are used extensively in [30, 31].

2.2.2 Riccati Form of Necessary Conditions

To begin, note that the form of (2.41a) is a particular case of a more general problem

$$Z = \min \frac{1}{2} x(t_f)^T F x(t_f) + \frac{1}{2} \int_0^{t_f} x^T Q x + x^T H \mu + \mu^T R \mu \, dt \tag{2.45a}$$

subject to

$$x' = Ax + Bv + M\mu \tag{2.45b}$$

where $F = 2\delta I$, ($\delta \geq 0$ and small), is symmetric positive semi-definite. The $J_v(\beta)$ problem is the case in which $F = 0$, but for the following derivation we leave the F term in the cost. A nonzero F matrix is also used in Section 2.4.5. Fixing the initial state, $x(0) = \xi$, and leaving $x(t_f)$ free, we minimize over all possible initial conditions to obtain an expression for the optimal cost which will be useful in one form of our algorithm.

Optimal Trajectory

Noting that extrema of (2.45) must also satisfy conditions (2.44) to be optimal, and that R is symmetric positive definite for $0 < \beta < 1$, we solve for μ in (2.44c) and substitute into (2.44a) and (2.44b) to obtain a system in (x, λ)

$$\begin{pmatrix} x \\ \lambda \end{pmatrix}' = \begin{bmatrix} A - \frac{1}{2} M R^{-1} H^T & -M R^{-1} M^T \\ \frac{1}{4} H R^{-1} H^T - Q & \frac{1}{2} H R^{-1} M^T - A^T \end{bmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} v.$$

Letting

$$\begin{aligned} S &= \tfrac{1}{2}MR^{-1}H^T \\ W &= \tfrac{1}{4}HR^{-1}H^T \text{ (symmetric,} \geq 0) \\ V &= MR^{-1}M^T \text{ (symmetric,} \geq 0) \end{aligned}$$

we can simplify the system to

$$\begin{pmatrix} x \\ \lambda \end{pmatrix}' = \begin{bmatrix} A-S & -V \\ W-Q & S^T-A^T \end{bmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} v \tag{2.46}$$

a system of $2n$ linear time invariant differential equations, with the vector $\begin{bmatrix} B \\ 0 \end{bmatrix} v$ acting as the forcing function. The system has the following boundary conditions

- At $t = 0$, n boundary conditions are provided by the initial conditions: $x(0) = \xi$.
- At $t = t_f$, n boundary conditions are provided by the transversality conditions:

  $$\lambda(t_f)^T = \frac{\partial}{\partial x(t_f)} \left[\frac{1}{2} x(t_f)^T F x(t_f) \right].$$

  Thus

  $$\lambda(t_f) = Fx(t_f). \tag{2.47}$$

To determine the form of the relationship between x and λ, let $\Omega(t;0)$ be the $2n \times 2n$ fundamental solution matrix for (2.46). Then

$$\begin{pmatrix} x(t) \\ \lambda(t) \end{pmatrix} = \Omega(t;0) \left\{ \begin{pmatrix} x(0) \\ \lambda(0) \end{pmatrix} + \int_0^t \Omega^{-1}(\tau;0) \begin{bmatrix} B \\ 0 \end{bmatrix} v \, d\tau \right\}.$$

Shifting this equation to the right end of the interval, we obtain

$$\begin{pmatrix} x(t_f) \\ \lambda(t_f) \end{pmatrix} = \Omega(t_f;t) \left\{ \begin{pmatrix} x(t) \\ \lambda(t) \end{pmatrix} + \int_t^{t_f} \Omega^{-1}(\tau;t) \begin{bmatrix} B \\ 0 \end{bmatrix} v \, d\tau \right\}.$$

Partitioning $\Omega(t_f;t)$ into four $n \times n$ blocks

$$\Omega(t_f;t) = \begin{bmatrix} \Omega_{11}(t_f;t) & \Omega_{12}(t_f;t) \\ \Omega_{21}(t_f;t) & \Omega_{22}(t_f;t) \end{bmatrix}$$

leads to

$$\begin{aligned} x(t_f) &= \Omega_{11}x(t) + \Omega_{12}\lambda(t) + \widehat{\Omega}\int_t^{t_f} \widehat{\widehat{\Omega}}^{-1}(\tau;t)Bv\;d\tau && (2.48a) \\ \lambda(t_f) &= \Omega_{21}x(t) + \Omega_{22}\lambda(t) && (2.48b) \end{aligned}$$

where $\widehat{\Omega}$ and $\widehat{\widehat{\Omega}}^{-1}$ are the parts of Ω and Ω^{-1} which conform to $x(t)$. Combining (2.47), (2.48a), and (2.48b)

$$\Omega_{21}x(t) + \Omega_{22}\lambda(t) = F\Omega_{11}x(t) + F\Omega_{12}\lambda(t) + F\widehat{\Omega}\int_t^{t_f} \widehat{\widehat{\Omega}}^{-1}(\tau;t)Bvd\tau$$

leads to an expression for $\lambda(t)$

$$\begin{aligned} \lambda(t) \;=\; & \left[\Omega_{22}(t_f;t) - F\Omega_{12}(t_f;t)\right]^{-1}\left[F\Omega_{11}(t_f;t) - \Omega_{21}(t_f;t)\right]x(t) \\ & + \left[\Omega_{22}(t_f;t) - F\Omega_{12}(t_f;t)\right]^{-1} F\widehat{\Omega}(t_f;t)\textstyle\int_t^{t_f} \widehat{\widehat{\Omega}}^{-1}(\tau;t)Bv\;d\tau \end{aligned} \qquad (2.49)$$

provided the indicated inverse exists. Note that at $t = t_f$, we know that $\Omega(t_f;t_f) = I$, i.e.,

$$\begin{array}{c} \Omega_{11}(t_f;t_f) = \Omega_{22}(t_f;t_f) = I \\ \Omega_{12}(t_f;t_f) = \Omega_{21}(t_f;t_f) = 0 \end{array}.$$

Thus $\Omega_{22}(t_f;t_f) - F\Omega_{12}(t_f;t_f) = I$ is nonsingular. Also $F\Omega_{11}(t_f;t_f) - \Omega_{21}(t_f;t_f) = F$ so that

$$\lambda(t_f) = IFx(t_f) + IFI0 = Fx(t_f)$$

in agreement with (2.47). Kalman, et al., [20] have shown that the inverse exists for all $t_0 \leq t < t_f$, so (2.49) is valid.

The form of (2.49) leads us to believe that $x(t)$ and $\lambda(t)$ are related by

$$\lambda(t) = K(t)x(t) - g(t). \tag{2.50}$$

From (2.46), we know that

$$x' = (A - S)x - V\lambda + Bv.$$

Combining this equation with (2.50) results in the expression of the optimal trajectory

$$x' = (A - S - VK)x + Vg + Bv. \tag{2.51}$$

Riccati Equation

Differentiating (2.50), we obtain

$$\lambda' = K'x + Kx' - g'. \tag{2.52}$$

Substituting (2.51) into (2.52), we obtain

$$\lambda' = [K' + K(A - S) - KVK]x + KVg + KBv - g'. \tag{2.53}$$

From (2.46), we also know that

$$\lambda' = (W - Q)x + (S^T - A^T)\lambda.$$

Combining this equation with (2.50) results in

$$\lambda' = [(S^T - A^T)K + W - Q]x + (A^T - S^T)g. \tag{2.54}$$

As long as an optimal solution exists, (2.53) and (2.54) must hold for all x and t. Equating coefficients yields

$$K' = (S^T - A^T)K + K(S - A) + KVK + W - Q \tag{2.55}$$

and

$$g' = (S^T - A^T + KV)g + KBv. \tag{2.56}$$

The boundary conditions for these equations are obtained at $t = t_f$ as follows:

- From (2.50), $\lambda(t_f) = K(t_f)x(t_f) - g(t_f)$,
- From (2.47), $\lambda(t_f) = Fx(t_f)$.

Both of these must hold for all $x(t_f)$, so

$$\begin{aligned} &K(t_f) = F \text{ [in agreement with the note below (2.49)]} \\ &g(t_f) = 0. \end{aligned}$$

With the boundary conditions completely specified at $t = t_f$, (2.55) and (2.56) can be solved to obtain $K(t)$ and $g(t)$ uniquely for all $t \in [0, t_f]$. Furthermore, if $K(t)$ is the solution of (2.55), and $K(t_f) = F$, then $K(t)$ is symmetric for all $t \in [0, t_f]$, because

$$(K')^T = [(S^T - A^T)K + K(S - A) + KVK + W - Q]^T$$

and thus

$$(K^T)' = K^T(S - A) + (S^T - A^T)K^T + K^TVK^T + W - Q$$

since W and Q are symmetric, which means that K and K^T are solutions of the same Riccati equation. This fact, combined with the boundary condition $K(t_f) = F = K(t_f)^T$, and the uniqueness property of ODE solutions, implies that $K = K^T$.

In fact, K is also positive definite and bounded for all $t \in [0, t_f]$. To see these facts, we must first show $W - Q$ to be negative definite. Recall the construction of the M matrix

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

where

$$M_i = \begin{bmatrix} \overline{M}_i & \widetilde{M}_i \end{bmatrix}, \quad N_i = \begin{bmatrix} \overline{N}_i & 0 \end{bmatrix}.$$

We have assumed that each M_i is full row rank, and that the coordinate change is done such that $\overline{N}_i$ is invertible. This implies that M is full row rank, and thus $V = MR^{-1}M^T$ is symmetric positive definite, since R is symmetric positive definite. Letting $C = \begin{bmatrix} \overline{N}_0^{-1} C_0 & -\overline{N}_0^{-1} C_1 \end{bmatrix}$, we can rewrite Q and H as

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

$$H = -4\beta C^T \overline{N}_0^{-1} \begin{bmatrix} 0 & \overline{N}_1 & 0 \end{bmatrix}.$$

Thus

$$\begin{aligned} W &= \tfrac{1}{4} H R^{-1} H^T \\ &= \tfrac{1}{4}(16\beta^2) C^T \overline{N}_0^{-1} \begin{bmatrix} 0 & \overline{N}_1 & 0 \end{bmatrix} R^{-1} \begin{bmatrix} 0 & \overline{N}_1 & 0 \end{bmatrix}^T \overline{N}_0^{-T} C \\ &= 2\beta^2 C^T \overline{N}_0^{-1} \overline{N}_1 \left[(1-\beta) I + \beta \overline{N}_1^T \overline{N}_0^{-T} \overline{N}_0^{-1} \overline{N}_1 \right]^{-1} \overline{N}_1^T \overline{N}_0^{-T} C. \end{aligned}$$

Performing a singular value decomposition on $\overline{N}_0^{-1} \overline{N}_1$, we obtain

$$\overline{N}_0^{-1} \overline{N}_1 = \widetilde{U} \widetilde{\Sigma} \widetilde{V} = \widetilde{U} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_m \end{bmatrix} \widetilde{V}$$

where, since $(\overline{N}_0^{-1} \overline{N}_1)^{-1}$ exists, all σ_i are bounded away from zero. Then

$$\begin{aligned} W &= 2\beta^2 C^T \widetilde{U} \widetilde{\Sigma} \widetilde{V} \left[(1-\beta) I + \beta (\widetilde{U} \widetilde{\Sigma} \widetilde{V})^T (\widetilde{U} \widetilde{\Sigma} \widetilde{V}) \right]^{-1} (\widetilde{U} \widetilde{\Sigma} \widetilde{V})^T C \\ &= 2\beta^2 C^T \widetilde{U} \widetilde{\Sigma} \left[(1-\beta) I + \beta \widetilde{\Sigma}^2 \right]^{-1} \widetilde{\Sigma} \widetilde{U}^T C \\ &= 2 C^T \widetilde{U} \begin{bmatrix} \frac{\beta^2 \sigma_1^2}{(1-\beta)+\beta\sigma_1^2} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{\beta^2 \sigma_m^2}{(1-\beta)+\beta\sigma_m^2} \end{bmatrix} \widetilde{U}^T C \end{aligned}$$

and

$$Q = 2\beta C^T C = 2\beta C^T \widetilde{U}\widetilde{U}^T C.$$

Subtracting, we obtain

$$\begin{aligned} W - Q &= 2C^T\widetilde{U}\begin{bmatrix} \ddots & & \\ & \frac{\beta^2\sigma_i^2}{(1-\beta)+\beta\sigma_i^2} & \\ & & \ddots \end{bmatrix}\widetilde{U}^T C - 2C^T\widetilde{U}\beta I\widetilde{U}^T C \\ &= 2C^T\widetilde{U}\begin{bmatrix} \ddots & & \\ & \frac{\beta^2\sigma_i^2}{(1-\beta)+\beta\sigma_i^2} - \beta & \\ & & \ddots \end{bmatrix}\widetilde{U}^T C. \end{aligned}$$

At $\beta = 1$, $\frac{\beta^2\sigma^2}{(1-\beta)+\beta\sigma^2} - \beta = 0$, and at $\beta = 0$, $\frac{\beta^2\sigma^2}{(1-\beta)+\beta\sigma^2} - \beta = 0$. This expression is quadratic in β, and so it can have at most two distinct roots. Thus, for values of β between 0 and 1, it will either be positive, or negative (or identically zero, which it is not), but not a combination of both. Evaluation at the midpoint of the interval shows that the expression is always negative. This fact implies that $W - Q$ is negative definite.

Now in solving the Riccati equation (2.55) in backward time

$$K' = (S^T - A^T)K + K(S - A) + KVK + W - Q$$

at $t = t_f$, we know that $K(t_f) = F = 2\delta I$, where δ is nonnegative and small. Thus

$$K'(t_f) = (\mathit{small}) + (\mathit{small}) + (\mathit{small})^2 + (\mathit{negative\ definite}) = (\mathit{negative\ definite})$$

and thus

$$K(t_f - \Delta t) > 0.$$

If $\delta = 0$, the same result holds. Thus we see that if F is symmetric positive semi-definite, then $K(t)$ is symmetric positive definite at $t = t_f - \Delta t$. Either $K(t)$ remains positive definite in backward time or it does not. If it remains positive definite, then at those times, τ, when it begins to lose definiteness, $\chi^T K(\tau-\Delta t)\chi > 0$ for all nonzero vectors χ. If $K(t)$ does not remain positive definite, then at those times, τ, when $K(\tau)$ loses definiteness there must exist a nonzero vector χ, for which $K(\tau)\chi = 0$. If this is true,

$$\begin{aligned}\chi^T K'(\tau)\chi &= \chi^T(S^T - A^T)K(\tau)\chi + \chi^T K(\tau)(S-A)\chi + \\ &\quad \chi^T K(\tau)VK(\tau)\chi + \chi^T(W-Q)\chi \\ &= 0 + 0 + 0 + (\textit{negative definite}).\end{aligned}$$

That is, $\chi^T K'(\tau)\chi < 0$, which implies that $\chi^T K(\tau - \Delta t)\chi > 0$, and $K(\tau - \Delta t)$ becomes positive definite again. Since this true for any vector that causes loss of positive definiteness as well as those vectors that do not, it is true for all vectors. Thus $K(t)$ remains positive definite.

Furthermore, at $t = \tau + \Delta t$, for any nonzero vector χ, for which $K(\tau)\chi = 0$,

$$\chi^T K'(\tau + \Delta t)\chi = (\textit{small}) + (\textit{small}) + (\textit{small})^2 + (\textit{negative definite})$$

which implies $\chi^T K'(\tau+\Delta t)\chi < 0$. By the same argument then, $K(\tau)$ cannot continue to lose definiteness, and is therefore bounded below, away from zero.

To see that K is bounded above, multiply (2.55) by K^{-1} on both sides.

$$K^{-1}K'K^{-1} = K^{-1}(S^T - A^T) + (S-A)K^{-1} + V + K^{-1}(W-Q)K^{-1}.$$

Since $(K^{-1})' = -K^{-1}K'K^{-1}$

$$(K^{-1})' = K^{-1}(A^T - S^T) + (A-S)K^{-1} + K^{-1}(Q-W)K^{-1} + (-V)$$

which, since $Q - W > 0$ and $(-V) < 0$, is the same Riccati equation as (2.55), with K^{-1} as the solution. So, by the same argument as that above for $K(t)$, $K(t)^{-1}$ is bounded away from zero. This fact implies that $K(t)$ is bounded above as well. Thus, if F is symmetric positive semi-definite, then $K(t)$ is the symmetric positive definite solution to a matrix Riccati equation, bounded above for all $t \in [0, t_f)$.

Optimal Cost

Returning to the cost function, suppose the optimal cost-to-go at time t has the form

$$Z^*(\beta, t) = \frac{1}{2}x(t)^T K(t)x(t) - g(t)^T x(t) + \varphi(t). \tag{2.57}$$

To obtain an expression for $\varphi(t)$, define the Hamilton-Jacobi equation for the system as

$$\begin{aligned}&\tfrac{\partial}{\partial t}Z^*(\beta,t) + \min_\mu \left[\tfrac{1}{2}x^T Qx + \tfrac{1}{2}x^T H\mu + \tfrac{1}{2}\mu^T R\mu\right.\\ &\left.+(Ax)^T \tfrac{\partial}{\partial x}Z^*(\beta,t) + (Bv)^T \tfrac{\partial}{\partial x}Z^*(\beta,t) + (M\mu)^T \tfrac{\partial}{\partial x}Z^*(\beta,t)\right] = 0.\end{aligned} \tag{2.58}$$

Letting $\lambda(t) = \frac{\partial}{\partial x}Z^*(\beta, t)$ be the Lagrange multiplier as before, the μ that minimizes the expression in brackets is given by (2.44c). Substituting for μ in (2.58) and expanding, we obtain

$$\frac{\partial}{\partial t}Z^* + \frac{1}{2}x^T(Q - W)x + x^T(A^T - S^T)\lambda - \frac{1}{2}\lambda^T V\lambda + v^T B\lambda = 0 \tag{2.59}$$

where S, W, and V are as before. Now, if (2.57) is correct, then

$$Z^* = \frac{1}{2}x^T Kx - g^T x + \varphi.$$

Thus

$$\frac{\partial}{\partial t}Z^* = \frac{1}{2}x^T K' x - g'^T x + \varphi'$$

and

$$\frac{\partial}{\partial x}Z^* = \lambda = Kx - g.$$

Substituting these into (2.59), then expanding and combining like terms we obtain

$$\begin{aligned}\tfrac{1}{2}x^T\left[K'+Q-W+(A^T-S^T)K+K(A-S)-KVK\right]x&\\ +x^T\left[-g'+(S^T-A^T+KV)g+KBv\right]&\\ +\varphi'-\tfrac{1}{2}g^TVg-v^TB^Tg &= 0.\end{aligned} \tag{2.60}$$

For (2.60) to hold for all x and t, (2.55) and (2.56) must hold, and

$$\varphi' = \frac{1}{2}g^TVg + v^TB^Tg \tag{2.61}$$

must also hold. Boundary conditions for φ, obtained from (2.57) at $t=t_f$, are

$$Z^*(\beta,t_f) = \frac{1}{2}x(t_f)^TK(t_f)x(t_f) - g(t_f)^Tx(t_f) + \varphi(t_f).$$

Since $K(t_f) = F$, and $g(t_f) = 0$, this expression reduces to the terminal cost, $\frac{1}{2}x(t_f)^TFx(t_f)$, if $\varphi(t_f)=0$.

To obtain an expression for the total optimal cost, first note that the total cost for the fixed initial condition problem is

$$Z^*(\beta,0) = \frac{1}{2}\xi^TK(0)\xi - g(0)^T\xi + \varphi(0).$$

The solution for the free initial state problem is obtained by letting ξ vary, and minimizing Z^* over the initial states. We find the necessary condition for a minimum is

$$\frac{\partial}{\partial\xi}Z^* = \xi^TK(0) - g(0)^T = 0.$$

Note that from (2.50)

$$\lambda(0)^T = x(0)^TK(0) - g(0)^T = \xi^TK(0) - g(0)^T.$$

Thus the optimal Z^* for the free-endpoint problem occurs when $\lambda(0)=0$

$$\begin{aligned} Z^*(\beta) &= \tfrac{1}{2}\xi^TK(0)\xi - g(0)^T\xi + \varphi(0)\\ &= [\xi^TK(0) - g(0)^T]\xi - \tfrac{1}{2}\xi^TK(0)\xi + \varphi(0)\\ &= \varphi(0) - \tfrac{1}{2}x(0)^TK(0)x(0)\end{aligned}$$

and since

$$\lambda(0) = K(0)x(0) - g(0) = 0 \tag{2.62}$$

then $K(0)x(0) = g(0)$, and we obtain the expression for the optimal cost

$$Z^*(\beta) = \varphi(0) - \frac{1}{2}x(0)^T g(0). \tag{2.63}$$

Summary

The necessary conditions for an optimal solution to the Z problem, (2.45), can be represented in terms of a stable matrix Riccati differential equation and two linear vector differential equations. The above derivation is completely consistent with the case $F = 0$. Thus, the $J_v(\beta)$ problem can be characterized in terms of these derived quantities. In the next section, we will formulate the max min problem in terms of the two forms of the necessary conditions derived above.

2.2.3 Problem Formulation in Terms of the Necessary Conditions

The previous discussion can be summarized by expressing (2.39) and (2.40) as boundary value problems (BVPs) in terms of the two forms of the necessary conditions: the raw form, (2.44), and the Riccati form, (2.55), (2.56), (2.51), (2.61), and (2.62).

Using the raw form of the necessary conditions, Version One, (2.39), becomes

$$(\gamma^*)^2 = \max_{v,\beta} Z(t_f) \tag{2.64a}$$

subject to the constraints

$$\begin{aligned}
x' &= Ax + Bv + M\mu && \text{(2.64b)}\\
\lambda' &= -Qx - \frac{1}{2}H\mu - A^T\lambda, \;\; \lambda(0) = \lambda(t_f) = 0 && \text{(2.64c)}\\
\theta' &= v^T v, \;\; \theta(0) = 0, \;\; \theta(t_f) = 1 && \text{(2.64d)}\\
Z' &= \frac{1}{2}\left[x^T Q x + x^T H\mu + \mu^T R\mu\right], \;\; Z(0) = 0 && \text{(2.64e)}\\
0 &= R\mu + \frac{1}{2}H^T x + M^T\lambda && \text{(2.64f)}\\
& \quad 0 < \beta < 1. && \text{(2.64g)}
\end{aligned}$$

Equations (2.64) are an index one BVP in $(x, \lambda, \theta, Z, \mu)$ since $R > 0$. Version Two, (2.40), becomes

$$(\gamma^*)^{-2} = \min_{v,\beta} \int_0^{t_f} \|v\|^2 dt \qquad \text{(2.65a)}$$

subject to the constraints

$$\begin{aligned}
x' &= Ax + Bv + M\mu && \text{(2.65b)}\\
\lambda' &= -Qx - \frac{1}{2}H\mu - A^T\lambda, \;\; \lambda(0) = \lambda(t_f) = 0 && \text{(2.65c)}\\
Z' &= \frac{1}{2}\left[x^T Q x + x^T H\mu + \mu^T R\mu\right], \;\; Z(0) = 0, \;\; Z(t_f) \geq 1 && \text{(2.65d)}\\
0 &= R\mu + \frac{1}{2}H^T x + M^T\lambda && \text{(2.65e)}\\
& \quad 0 < \beta < 1. && \text{(2.65f)}
\end{aligned}$$

Using the Riccati form of the necessary conditions, Version One, (2.39), becomes

$$(\gamma^*)^2 = \max_{v,\beta} \left[\varphi(0) - \frac{1}{2}x(0)^T g(0)\right] \qquad \text{(2.66a)}$$

subject to the constraints

$$\begin{aligned}
K' &= (S^T - A^T)K + K(S-A) + KVK + W - Q, \quad K(t_f) = 0 && (2.66\text{b})\\
g' &= (S^T - A^T + KV)g + KBv, \quad g(t_f) = 0 && (2.66\text{c})\\
x' &= (A - S - VK)x + Vg + Bv && (2.66\text{d})\\
\varphi' &= \frac{1}{2}g^TVg + g^TBv, \quad \varphi(t_f) = 0 && (2.66\text{e})\\
\theta' &= v^Tv, \quad \theta(0) = 0, \quad \theta(t_f) = 1 && (2.66\text{f})\\
0 &= K(0)x(0) - g(0) && (2.66\text{g})\\
& 0 < \beta < 1. && (2.66\text{h})
\end{aligned}$$

Version Two, (2.40), becomes

$$(\gamma^*)^{-2} = \min_{v,\beta} \int_0^{t_f} \|v\|^2 dt \qquad (2.67\text{a})$$

subject to the constraints

$$\begin{aligned}
K' &= (S^T - A^T)K + K(S-A) + KVK + W - Q, \quad K(t_f) = 0 && (2.67\text{b})\\
g' &= (S^T - A^T + KV)g + KBv, \quad g(t_f) = 0 && (2.67\text{c})\\
x' &= (A - S - VK)x + Vg + Bv && (2.67\text{d})\\
\varphi' &= \frac{1}{2}g^TVg + g^TBv, \quad \varphi(t_f) = 0 && (2.67\text{e})\\
0 &= K(0)x(0) - g(0) && (2.67\text{f})\\
1 &\leq \varphi(0) - \frac{1}{2}x(0)^Tg(0) && (2.67\text{g})\\
& 0 < \beta < 1. && (2.67\text{h})
\end{aligned}$$

Equations (2.64e) and (2.65d) define $J_v(\beta)$ explicitly, while (2.67g) and the right hand side of (2.66a) define it in terms of (2.63). Equations (2.65d) and (2.67g) also reflect the requirement for v to be proper, $J_v(\beta) \geq 1$. Note that both formulations of Version One, (2.64) and (2.66), require the detection signal to be normalized. Thus, for these problems, the minimum energy proper detection signal must be rescaled by $\frac{1}{\gamma^*}$. No rescaling is necessary for the formulations of Version Two, (2.65) and (2.67).

2.2.4 Sufficient Conditions

The arguments of the two previous sections relied on the assumption that the necessary conditions guaranteed a minimum. Sufficient conditions for a local minimum are in fact included in the necessary conditions as well. To see this fact, recall that the Hamiltonian for our problem is

$$\widehat{H} = \frac{1}{2}x^TQx + \frac{1}{2}x^TH\mu + \frac{1}{2}\mu^TR\mu + \lambda^T(Ax + Bv + M\mu). \tag{2.68}$$

It is a well known fact (Athans and Falb [1], for example), that if the matrix

$$\begin{bmatrix} \frac{\partial^2 \widehat{H}}{\partial x^2} & \frac{\partial^2 \widehat{H}}{\partial x\, \partial \mu} \\ \frac{\partial^2 \widehat{H}}{\partial \mu\, \partial x} & \frac{\partial^2 \widehat{H}}{\partial \mu^2} \end{bmatrix} \tag{2.69}$$

is positive semi-definite, then the noise, μ, which satisfies

$$\widehat{H}_\mu = 0 \tag{2.70}$$

(one of the Euler equations) is at least locally optimal. From (2.68), we find that

$$\begin{aligned} \frac{\partial \widehat{H}}{\partial x} &= Qx + \frac{1}{2}H\mu + A^T\lambda && (2.71a)\\ \frac{\partial^2 \widehat{H}}{\partial x^2} &= Q && (2.71b)\\ \frac{\partial \widehat{H}}{\partial \mu} &= R\mu + \frac{1}{2}H^Tx + M^T\lambda && (2.71c)\\ \frac{\partial^2 \widehat{H}}{\partial \mu^2} &= R && (2.71d)\\ \frac{\partial^2 \widehat{H}}{\partial x\, \partial \mu} &= \frac{1}{2}H && (2.71e)\\ \frac{\partial^2 \widehat{H}}{\partial \mu\, \partial x} &= \frac{1}{2}H^T. && (2.71f) \end{aligned}$$

Substituting (2.71) into (2.69), we obtain

$$\begin{bmatrix} Q & \frac{1}{2}H \\ \frac{1}{2}H^T & R \end{bmatrix} \tag{2.72}$$

which is equivalent to G^TG, a symmetric positive semi-definite matrix, where

$$G = \begin{bmatrix} \sqrt{2\beta}\overline{N}_0^{-1}C_0 & -\sqrt{2\beta}\overline{N}_0^{-1}C_1 & 0 & -\sqrt{2\beta}\overline{N}_0^{-1}\overline{N}_1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{2\beta}I & 0 & 0 \\ 0 & 0 & 0 & \sqrt{2(1-\beta)}I & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2(1-\beta)}I \end{bmatrix} \tag{2.73}$$

Thus (2.72) is positive semi-definite. Since the higher derivatives of $\widehat{H}$ are zero, the noise, μ, obtained from (2.70) and (2.71c) minimizes, at least locally, the cost [1].

2.3 The Minimum Energy Detection Signal Algorithm

It is useful at this point to put the detection signal problem in algorithm form. The algorithm described below is suitable for solving in Boeing's Sparse Optimal Control Software (SOCS) [3, 4]. The SOCS package is a collection of FORTRAN 77 subroutines capable of solving a great variety of optimal control problems. The package can handle many types of constraints, including L^2 and algebraic constraints, which are the types of constraints generated by the problems considered in this thesis. The main drawback of the software is that it is written in FORTRAN 77, which has limited matrix capability, and therefore requires translation of all system parameters into indexed variable form. This can be overcome by the use of additional software which can automatically generate FORTRAN code from matrix formatted input. In

Appendix A, we describe routines in MATLAB, by The MathWorks, Inc. [26] and MAPLE, by Waterloo Maple, Inc. [18] which can be used to accomplish the required translation, as well as a sample SOCS driver file.

The algorithm requires the system model to be in the form previously described

$$x_i' = A_i x_i + B_i v + M_i \mu_i \tag{2.74a}$$

$$y = C_i x_i + N_i \mu_i \tag{2.74b}$$

for $i = 0$ and 1, where x_i, y, v, and μ_i are the system states, output, detection signal, and noise, respectively. The matrices A_i, B_i, C_i, M_i, and N_i must be matrices of appropriate dimensions, with M_i and N_i having full row rank. In addition, $[A_i \;\; C_i]$ must be observable.

The minimum energy detection signal (MEDS) algorithm (with appropriate software in parentheses):

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

$$N_i^T = Q_i R_i \tag{2.75}$$

   where the Q_i are unitary matrices. Thus

$$N_i = R_i^T Q_i^T \tag{2.76}$$

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

   (a) Let $R_i^T = [\overline{N}_i \;\; 0]$, where $\overline{N}_i$ is invertible

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

$$N_i \mu_i = [\overline{N}_i \;\; 0] \begin{pmatrix} \overline{\mu}_i \\ \widetilde{\mu}_i \end{pmatrix} \tag{2.77}$$

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

$$M_i \mu_i = M_i Q_i^{-T} Q_i^T \mu_i = [\overline{M}_i \;\; \widetilde{M}_i] \begin{pmatrix} \overline{\mu}_i \\ \widetilde{\mu}_i \end{pmatrix} \tag{2.78}$$

Note the system model, (2.74), becomes

$$x_i' = A_i x_i + B_i v + \overline{M}_i \overline{\mu}_i + \widetilde{M}_i \widetilde{\mu}_i \tag{2.79a}$$

$$y = C_i x_i + \overline{N}_i \overline{\mu}_i \tag{2.79b}$$

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

(a) Combine both equations (2.79b) for y, solve for $\overline{\mu}_0$, and substitute into (2.79a), $i = 0$

$$x_0' = (A_0 - \overline{M}_0 \overline{N}_0^{-1} C_0) x_0 + \overline{M}_0 \overline{N}_0^{-1} C_1 x_1 + B_0 v + \widetilde{M}_0 \widetilde{\mu}_0 + \overline{M}_0 \overline{N}_0^{-1} \overline{N}_1 \overline{\mu}_1 \tag{2.80}$$

(b) Let

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

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

Note the system model, (2.79), is now

$$x' = Ax + Bv + M\mu \tag{2.81}$$

4. Compute new system matrices (MATLAB)

(a) Let $C = [\overline{N}_0^{-1}C_0 \quad -\overline{N}_0^{-1}C_1]$ and $N = \overline{N}_0^{-1}[0 \ \ \overline{N}_1 \ \ 0]$, with the columns of N conforming to μ

(b) Let $H = -4\beta C^T N$ and $Q = 2\beta C^T C$

(c) Let

$$R = 2\begin{bmatrix} \beta I & 0 & 0 \\ 0 & (1-\beta)I + \beta\overline{N}_1^T\overline{N}_0^{-T}\overline{N}_0^{-1}\overline{N}_1 & 0 \\ 0 & 0 & (1-\beta)I \end{bmatrix}$$

with its rows and columns conforming to μ

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

(e) For Riccati version, compute $S = \frac{1}{2}MR^{-1}H^T$, $W = \frac{1}{4}HR^{-1}H^T$, and $V = MR^{-1}M^T$

5. Perform constrained optimization (SOCS)

(a) Version One, solve

$$(\gamma^*)^2 = \max_{v,\beta} Z(t_f) \tag{2.82a}$$

subject to the constraints

$$\begin{aligned} x' &= Ax + Bv + M\mu && (2.82\text{b}) \\ \lambda' &= -Qx - \frac{1}{2}H\mu - A^T\lambda, \ \ \lambda(0) = \lambda(t_f) = 0 && (2.82\text{c}) \\ \theta' &= v^Tv, \ \ \theta(0) = 0, \ \ \theta(t_f) = 1 && (2.82\text{d}) \\ Z' &= \frac{1}{2}\left[x^TQx + x^TH\mu + \mu^TR\mu\right], \ \ Z(0) = 0 && (2.82\text{e}) \\ 0 &= R\mu + \frac{1}{2}H^Tx + M^T\lambda && (2.82\text{f}) \\ & 0.01 \le \beta \le 0.99 && (2.82\text{g}) \end{aligned}$$

(b) Version Two, solve

$$(\gamma^*)^{-2} = \min_{v,\beta} \int_0^{t_f} \|v\|^2 dt \tag{2.83a}$$

subject to the constraints

$$x' = Ax + Bv + M\mu \tag{2.83b}$$

$$\lambda' = -Qx - \frac{1}{2}H\mu - A^T\lambda, \quad \lambda(0) = \lambda(t_f) = 0 \tag{2.83c}$$

$$Z' = \frac{1}{2}\left[x^TQx + x^TH\mu + \mu^TR\mu\right], \quad Z(0) = 0, \quad Z(t_f) \geq 1 \tag{2.83d}$$

$$0 = R\mu + \frac{1}{2}H^Tx + M^T\lambda \tag{2.83e}$$

$$0.01 \leq \beta \leq 0.99 \tag{2.83f}$$

(c) Version One, the Riccati form, solve

$$(\gamma^*)^2 = \max_{v,\beta} \left[\varphi(0) - \frac{1}{2}x(0)^Tg(0)\right] \tag{2.84a}$$

subject to the constraints

$$K' = (S^T - A^T)K + K(S - A) + KVK + W - Q \tag{2.84b}$$

$$g' = (S^T - A^T + KV)g + KBv, \quad g(t_f) = 0, \quad K(t_f) = 0 \tag{2.84c}$$

$$x' = (A - S - VK)x + Vg + Bv \tag{2.84d}$$

$$\varphi' = \frac{1}{2}g^TVg + g^TBv, \quad \varphi(t_f) = 0 \tag{2.84e}$$

$$\theta' = v^Tv, \quad \theta(0) = 0, \quad \theta(t_f) = 1 \tag{2.84f}$$

$$0 = K(0)x(0) - g(0) \tag{2.84g}$$

$$0.01 \leq \beta \leq 0.99 \tag{2.84h}$$

(d) Version Two, the Riccati form, solve

$$(\gamma^*)^{-2} = \min_{v,\beta} \int_0^{t_f} \|v\|^2 dt \tag{2.85a}$$

subject to the constraints

$$\begin{aligned}
K' &= (S^T - A^T)K + K(S-A) + KVK + W - Q && (2.85b)\\
g' &= (S^T - A^T + KV)g + KBv, \quad g(t_f) = 0, \quad K(t_f) = 0 && (2.85c)\\
x' &= (A - S - VK)x + Vg + Bv && (2.85d)\\
\varphi' &= \frac{1}{2}g^T Vg + g^T Bv, \quad \varphi(t_f) = 0 && (2.85e)\\
0 &= K(0)x(0) - g(0) && (2.85f)\\
1 &\leq \varphi(0) - \frac{1}{2}x(0)^T g(0) && (2.85g)\\
& \quad 0.01 \leq \beta \leq 0.99 && (2.85h)
\end{aligned}$$

For each formulation, v and μ are treated as control variables while β is passed to SOCS as a parameter. The constraint (2.85h) and the others identical to it are required to bound β away from 0 and 1 early in the optimization process. The minimum energy proper detection signal, $\widehat{v}$, is $\frac{v}{\gamma^*}$ for (2.82) and (2.84). The minimum energy proper detection signal, $\widehat{v}$, is simply v for (2.83) and (2.85).

2.4 Variations

The assumptions and problem formulations described above limit the types of problems that may be solved by the MEDS algorithm. This fact leads to questions about extending the algorithm to variations of the problem: Can the algorithm handle problems with more than one fault model? What if the problem must be formulated more in terms of the original system matrices? What happens when a control is already present in the model? Could alternative cost functions be minimized? Does knowledge of initial conditions impact the theory? This section will address these questions and demonstrate the flexibility of the algorithm.

It should be noted that while this section uses Equations (2.65) as the form of the model around which we describe variations, any of the four forms could be used. It is not our intent to promote one form of the algorithm over another, merely to present the options available.

2.4.1 Multiple Fault Models

Many, if not most, real world systems which are represented as multi-model control systems have more than one fault model. Unfortunately, this is one of the types of problems that approaches such as [30] are unable to efficiently address. However, our MEDS algorithm can solve the problem in one of two ways. For this discussion, assume that three models exist for the system: the nominal model, $(i = 0)$, and two fault models, $(i = 1, 2)$. Obviously, problems with more than two fault models may be solved with either method using combinatorial extensions.

The first method applies the MEDS algorithm in a sequential manner, an inefficient approach which [30] is forced to take. Dividing the test period, $[0, t_f]$, into three subperiods, $[0, \frac{t_f}{3}]$, $[\frac{t_f}{3}, \frac{2t_f}{3}]$, and $[\frac{2t_f}{3}, t_f]$, use the algorithm to solve model 0 versus model 1 in the first subperiod for v_{01}, model 0 versus model 2 in the second subperiod for v_{02}, and model 1 versus model 2 in the final subperiod for v_{12}. Because a detection signal which is proper on an interval is still proper on any longer interval including the original interval (due to the fact that an L^2 noise on the test period has the same or smaller norm on a shorter subperiod), the composite v obtained by applying v_{01} on $[0, \frac{t_f}{3}]$, then v_{02} on $[\frac{t_f}{3}, \frac{2t_f}{3}]$, and then v_{12} on $[\frac{2t_f}{3}, t_f]$, will be a proper detection signal. It will also be the minimum energy proper detection signal for the problem via this approach.

Since this method involves shorter test periods for the individual comparisons, it has a tendency to produce detection signals of higher than necessary energy. The second method avoids this tendency by simultaneously solving all three subproblems independently over the entire test period for a common v, an approach of which many prior methods, including [30], are incapable. Consider

$$(\gamma^*)^{-2} = \min_{v,\beta_{01},\beta_{02},\beta_{12}} \int_0^{t_f} \|v\|^2 dt \tag{2.86a}$$

subject to the constraints

$$\begin{aligned}
x_{01}' &= A_{01}x_{01} + B_{01}v + M_{01}\mu_{01} && (2.86b)\\
\lambda_{01}' &= -Q_{01}x_{01} - \frac{1}{2}H_{01}\mu_{01} - A_{01}^T\lambda_{01},\ \ \lambda_{01}(0) = \lambda_{01}(t_f) = 0 && (2.86c)\\
Z_{01}' &= \frac{1}{2}\left[x_{01}^TQ_{01}x_{01} + x_{01}^TH_{01}\mu_{01} + \mu_{01}^TR_{01}\mu_{01}\right],\ Z_{01}(0) = 0,\ Z_{01}(t_f) \geq 1 && (2.86d)\\
0 &= R_{01}\mu_{01} + \frac{1}{2}H_{01}^Tx_{01} + M_{01}^T\lambda_{01} && (2.86e)\\
& 0.01 \leq \beta_{01} \leq 0.99 && (2.86f)\\
x_{02}' &= A_{02}x_{02} + B_{02}v + M_{02}\mu_{02} && (2.86g)\\
\lambda_{02}' &= -Q_{02}x_{02} - \frac{1}{2}H_{02}\mu_{02} - A_{02}^T\lambda_{02},\ \ \lambda_{02}(0) = \lambda_{02}(t_f) = 0 && (2.86h)\\
Z_{02}' &= \frac{1}{2}\left[x_{02}^TQ_{02}x_{02} + x_{02}^TH_{02}\mu_{02} + \mu_{02}^TR_{02}\mu_{02}\right],\ Z_{02}(0) = 0,\ Z_{02}(t_f) \geq 1 && (2.86i)\\
0 &= R_{02}\mu_{02} + \frac{1}{2}H_{02}^Tx_{02} + M_{02}^T\lambda_{02} && (2.86j)\\
& 0.01 \leq \beta_{02} \leq 0.99 && (2.86k)\\
x_{12}' &= A_{12}x_{12} + B_{12}v + M_{12}\mu_{12} && (2.86l)\\
\lambda_{12}' &= -Q_{12}x_{12} - \frac{1}{2}H_{12}\mu_{12} - A_{12}^T\lambda_{12},\ \ \lambda_{12}(0) = \lambda_{12}(t_f) = 0 && (2.86m)\\
Z_{12}' &= \frac{1}{2}\left[x_{12}^TQ_{12}x_{12} + x_{12}^TH_{12}\mu_{12} + \mu_{12}^TR_{12}\mu_{12}\right],\ Z_{12}(0) = 0,\ Z_{12}(t_f) \geq 1 && (2.86n)\\
0 &= R_{12}\mu_{12} + \frac{1}{2}H_{12}^Tx_{12} + M_{12}^T\lambda_{12} && (2.86o)\\
& 0.01 \leq \beta_{12} \leq 0.99 && (2.86p)
\end{aligned}$$

where a vector$_{ij}$ or a matrix$_{ij}$ is the unsubscripted vector/matrix in the simple model (2.65) when model i and model j are the two models for which the problem is being formulated. Note that only the detection signal is common between models. This independence increases the size of the problem with both the size and number of models.

It should also be noted that if the goal is to detect a fault, but it is not important to distinguish between fault models, the detection signal should be minimized by solving only those pairwise problems including the nominal model (i.e., in the three model case, model 0 versus model 1, and model 0 versus model 2). The nominal model will then be distinguishable from the fault models, but individual fault model outputs may overlap each other. These cases will be discussed further in the next chapter. Examples of multiple fault model problems are included in Chapter 4.

2.4.2 Unreduced Model

In addition to multiple fault models, many system models have parameters which represent some kind of physical quantity. Model reduction hinders the ability to match parameters with real quantities, and so it may be useful in these cases to formulate the problem more in terms of original system equations. Consider the characterization of not proper, (2.10), after applying the result, (2.23)

$$\max_{0\leq\beta\leq 1} \min \int_0^{t_f} \beta|\mu_0(t)|^2 + (1-\beta)|\mu_1(t)|^2 \, dt < 1 \tag{2.87a}$$

with the minimum subject to

$$\begin{aligned} x_0' &= A_0x_0 + B_0v + M_0\mu_0 && (2.87b)\\ x_1' &= A_1x_1 + B_1v + M_1\mu_1 && (2.87c)\\ 0 &= C_0x_0 - C_1x_1 + N_0\mu_0 - N_1\mu_1. && (2.87d) \end{aligned}$$

Instead of reducing (2.87) as before, let

$$x = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu_0 \\ \mu_1 \end{pmatrix}, \quad V_\beta = 2\begin{bmatrix} \beta I & 0 \\ 0 & (1-\beta)I \end{bmatrix},$$

$$A = \begin{bmatrix} A_0 & 0 \\ 0 & A_1 \end{bmatrix}, \quad B = \begin{bmatrix} B_0 \\ B_1 \end{bmatrix}, \quad M = \begin{bmatrix} M_0 & 0 \\ 0 & M_1 \end{bmatrix}$$

$$C = [C_0 \quad -C_1], \quad N = [N_0 \quad -N_1].$$

Then the characterization of not proper, (2.87) becomes

$$\max_{0\leq\beta\leq 1} \min \frac{1}{2}\int_0^{t_f} \mu^T V_\beta \mu \, dt < 1 \tag{2.88a}$$

with the minimum subject to

$$x' = Ax + Bv + M\mu \tag{2.88b}$$

$$0 = Cx + N\mu. \tag{2.88c}$$

The Hamiltonian for the inner problem is

$$\widehat{H} = \frac{1}{2}\mu^T V_\beta \mu + \lambda_0^T(-x' + Ax + Bv + M\mu) + \lambda_1^T(Cx + N\mu). \tag{2.89}$$

Applying the necessary conditions, (2.43), to (2.89), we obtain

$$0 = V_\beta \mu + M^T\lambda_0 + N^T\lambda_1 \tag{2.90}$$

$$\lambda_0' = -A^T\lambda_0 - C^T\lambda_1 \tag{2.91}$$

as well as (2.88b)-(2.88c). Thus we obtain an unreduced problem that can be expressed in the same form as the reduced problem. In terms similar to (2.65)

$$(\gamma^*)^{-2} = \min_{v,\beta} \int_0^{t_f} \|v\|^2 dt \tag{2.92a}$$

subject to the constraints

$$x' = Ax + Bv + M\mu \tag{2.92b}$$

$$0 = Cx + N\mu \tag{2.92c}$$

$$\lambda_0' = -A^T\lambda_0 - C^T\lambda_1, \quad \lambda_0(0) = \lambda_0(t_f) = 0 \tag{2.92d}$$

$$Z' = \frac{1}{2}\mu^T V_\beta \mu, \quad Z(0) = 0, \quad Z(t_f) \geq 1 \tag{2.92e}$$

$$0 = V_\beta \mu + M^T\lambda_0 + N^T\lambda_1 \tag{2.92f}$$

$$0 < \beta < 1. \tag{2.92g}$$

For this formulation to be worthwhile, the benefit obtained from association of model parameters with quantities of interest should outweigh the impact of the increased dimension of the problem. In addition, (2.92b) and (2.92c) comprise an index one BVP, so optimization software used to solve it must have the capability to handle DAEs.

2.4.3 Controlled Systems

Even in cases for which a reduced model is acceptable, the presence of a known reference control, u, may appear to complicate matters. In actuality, controlled systems can easily be handled by the algorithm. By using the same input channels for the detection signal, v, and the control, u, the problem goes from

$$\min \|v\| \text{ subject to } \max_{0<\beta<1} J_v(\beta) \geq 1 \tag{2.93}$$

to

$$\min \|v\| \text{ subject to } \max_{0<\beta<1} J_{u+v}(\beta) \geq 1. \tag{2.94}$$

Thus instead of v appearing in the constraints, $u+v$ would appear. Note that in this case, it is possible that a zero detection signal may be the minimum proper. This would occur when the control by itself is already proper.

If the reference control does not come in on the same channels as the detection signal the differential equation becomes

$$x_i' = A_i x_i + B_i v + E_i u + M_i \mu_i. \tag{2.95}$$

By letting $w = [w_1, w_2] = [v, u]$, we can construct a problem similar to (2.94)

$$\min \|w_1\| \text{ subject to } \max_{0<\beta<1} J_w(\beta) \geq 1 \text{ and } w_2 = u. \tag{2.96}$$

An alternative approach to solving the controlled system is to eliminate u just as we eliminate the output, y, to obtain (2.13). Using the expression for the known control, model reduction is accomplished as described in Section 2.1.2.

2.4.4 Alternative Cost Functions

Another complication occurs when the energy of the detection signal is not the desired cost function. One possible alternative cost function is the power of the detection signal, with noise of bounded power. For this case, the noise model, (2.3), becomes

$$||\mu_i||^2 = \int_0^{t_f} |\mu_i(t)|^2 \, dt < Kt_f, \quad i = 0, 1\ , K > 0. \tag{2.97}$$

By retracing the problem development in the first part of this chaper, we see that the only differences between the bounded power noise/minimum power detection signal case and the bounded energy noise/minimum energy detection signal case are the values of the right hand sides of the inequalities (2.3) and (2.97), and the scaling of the objective function. Thus, for the bounded power/minimum power detection signal problem, (2.65) becomes

$$\min_{v,\beta} \frac{1}{Kt_f} \int_0^{t_f} ||v||^2 dt \tag{2.98a}$$

subject to the constraints

$$\begin{aligned}
x' &= Ax + Bv + M\mu && \text{(2.98b)}\\
\lambda' &= -Qx - \frac{1}{2}H\mu - A^T\lambda, \;\; \lambda(0) = \lambda(t_f) = 0 && \text{(2.98c)}\\
Z' &= \frac{1}{2}\left[x^TQx + x^TH\mu + \mu^TR\mu\right], \;\; Z(0) = 0, \;\; Z(t_f) \geq Kt_f && \text{(2.98d)}\\
0 &= R\mu + \frac{1}{2}H^Tx + M^T\lambda && \text{(2.98e)}\\
& \quad 0 < \beta < 1. && \text{(2.98f)}
\end{aligned}$$

Other alternative problems that introduce complications no more difficult than those above include

- bounded power noise/minimum energy detection signal,
- bounded power noise/minimum of a function of the detection signal ($Qx+Rv$),
- bounded energy noise/minimum of a function of the detection signal ($Qx+Rv$).

Each of the problems in this list is easily constructed and efficiently handled by SOCS.

2.4.5 Knowledge of Initial Conditions

As a final variation on the problem, consider the case in which a weighted initial condition is added to the noise constraint. In this case, instead of the bounded energy noise model, (2.3), we have

$$\mathcal{S}_i(x_i(0),\mu_i) \equiv x_i(0)^T P_{i,0}^{-1} x_i(0) + \int_0^{t_f} |\mu_i(t)|^2 dt < 1, \quad i = 0,1 \tag{2.99}$$

where the $x_i(0)$ are the initial states and the $P_{i,0}^{-1}$ are the weight matrices. This can also be generalized to

$$\mathcal{S}_{a_i,i}(x_i(0),\mu_i) \equiv [x_i(0)-a_i]^T P_{i,0}^{-1}[x_i(0)-a_i] + \int_0^{t_f} |\mu_i(t)|^2 dt < 1, \quad i = 0,1 \tag{2.100}$$

for some fixed vector a_i. (2.99) is the case addressed in [30], and this formulation leads to quite a different method. The approach is made possible by the elimination of the long axis in each output set caused by the free initial condition. Nikoukhah, et al. [30] use standard Kalman filtering combined with extensive Riccati differential equation theory to derive an elegant method for computing the minimum energy detection signal and the separability index. One chooses a γ, then solves a Riccati equation until it diverges. If the divergence occurs inside the test period, increase γ and repeat. If the Riccati equation diverges past the end of the test period, decrease γ and repeat. Using a bisection method, γ^* can be found with any desired accuracy. Once γ^* is fixed, the minimum energy detection signal is computed by solving a two-point BVP via two other Riccati equations. Thus, whereas our approach is direct, relying on the basics of optimal control theory, the approach of [30] is recursive in nature, relying on the stability of Riccati matrices. Unfortunately, the approach of [30] is limited to linear systems of only two models.

Our approach, however is not only extendable to certain nonlinear systems we discuss in Chapter 5, it can handle systems of more than two models as previously mentioned, and it can handle the noise models (2.99) and (2.100). In fact, the theory for those noise models is actually slightly simpler due to the elimination of the long axis in the output sets. $P(x, \mu, \beta)$ from (2.18) becomes

$$P(x,\mu,\beta) = \beta x_0(0)^T P_{0,0}^{-1} x_0(0) + (1-\beta) x_1(0)^T P_{1,0}^{-1} x_1(0) + \frac{1}{2}\int_0^{t_f} x^T Q x + x^T H \mu + \mu^T R \mu \, dt \quad (2.101)$$

but the new term is positive as long as $P_{i,0} > 0$. Thus, $J_v(\beta) = \min P(x, \mu, \beta) \geq 1$ still characterizes a proper detection signal. While the boundary conditions change slightly, the new term involving the initial conditions does not appear in the variational equations. Thus the problem is virtually the same, and we expect the impact of the weight matrices to diminish as the test period increases in length. SOCS should encounter no difficulties from this variation.

2.4.6 Conclusion

While other variations of the basic problem exist which may be more difficult to accomodate, it is clear from the cases discussed in this section that the algorithm is quite flexible and can be adapted to solve many types of multi-model problems. In fact, current theory for the multiple fault model case only provides for problem development; it does not provide a practical means with which to solve the problem. Our MEDS algorithm not only provides theoretical development for the multiple fault model case, it also supplies a simple method for solving such problems.

Chapter 3

Model Identification via the Separating Hyperplane

3.1 The Problem - Determining the Origin of a Given Output

As was shown in the previous chapter, the MEDS algorithm guarantees that the output sets, $\mathcal{A}^i(v)$, from the two possible system models are disjoint. In other words, given an output from the system, it is the result of one model or the other, but not both. The algorithm does not, however, tell us from which model the output is derived. In this chapter, we will address the model identification step, and by specifying the correct model for a given output using a separating hyperplane, we will complete the development of our multi-model approach to fault detection and model identification in linear descriptor systems.

3.1.1 Problem Setup

Recall that the true model of the system is one of two models

$$x_i' = A_i x_i + B_i \widehat{v} + M_i \mu_i \tag{3.1a}$$

$$y_i = C_i x_i + N_i \mu_i \tag{3.1b}$$

for $i = 0$ and 1, where all variables are as previously defined except $\widehat{v}$, which is now the minimum energy detection signal from the MEDS algorithm. In addition, with the application of $\widehat{v}$, each model has a distinct output, and so the output, y, is now subscripted.

Recall also from Section 2.1.1, that the output sets, $\mathcal{A}^i(v)$, are open convex sets. The sets are open because $\|\mu_i\| < 1$ and the N_i are full row rank. In effect, the noise contribution to an output set can be likened to the addition of an open "ball" (in the L^2 sense) to the boundaries of the noiseless output sets. The fact that the minimum energy detection signal is being applied implies that while the output sets are disjoint, their closures share at least one common point at any given time. This occurs due to the definition of *minimum* proper. Suppose a detection signal which is infinitesimally "smaller" than the minimum energy detection signal is applied. In this case, the detection signal would not be proper, and the output sets would not be disjoint. In order for the open output sets to intersect with the application of the infinitesimally smaller detection signal, the closures of the output sets must intersect with the application of the minimum energy proper detection signal.

Assuming a unique point of intersection of the closures of the output sets, which occurs when at least one of the sets is strictly convex, that point is easily computed. Recall from the previous chapter that the output equations were combined in order to reduce the dimension of the model. That is

$$y = C_0 x_0 + N_0 \mu_0 \tag{3.2a}$$

$$y = C_1 x_1 + N_1 \mu_1 \tag{3.2b}$$

were equated to each other to eliminate y. When the optimal trajectory, x_1, and the "optimal" noise, μ_1, (from the optimal solution to the MEDS algorithm) are substituted into equation (3.2b), the resulting y is the point of intersection of the closures of the $\mathcal{A}^i(v)$. This is true because the MEDS algorithm actually makes use of the complements of the output sets in the optimization, finding the detection signal of smallest norm that does not satisfy the not proper conditions, as discussed in the previous chapter. The complements of the output sets are closed sets in L^2, and the y that results from inserting the solution of the MEDS algorithm into (3.2) is on the shared boundary of these sets. Note that we could use either equation of (3.2) to compute y, but in the model reduction, part of μ_0 is eliminated; all of μ_1 is available from the optimal solution. For the remainder of this discussion, let the common y be called $\overline{y}$.

Problems may result when the closures of the output sets intersect at more than one point at any given time. Non-unique intersections among convex sets can only occur when the sets have flat, parallel sides along their common border, i.e., both sets are not strictly convex. This geometry is present in cases where both A_i matrices share a common eigenvalue, eigenvector pair. While this occurrence is rare, since perturbing an element of a matrix almost always changes all of the eigenvalues of that matrix, software packages may fail in its presence (due to a non-unique optimal solution), and thus we should be prepared for it. To counteract its effect, we simply project y by multiplying both output equations by a matrix that eliminates the parallel part. This operation will result in lower dimensional output sets which have no flat, parallel sides. In the remainder of our discussion we will assume that such a projection has been applied and the resulting lower dimensional output sets are strictly convex.

3.1.2 The Separating Hyperplane

A convenient feature of convex sets is the *separating* hyperplane [34]. Given two disjoint convex sets, there exists a hyperplane that separates the sets; that is, one set is above the hyperplane while the other set is below. Such a hyperplane exists for the two output sets $\mathcal{A}^i(v)$, and it contains the point $\overline{y}$. In fact, because the corners of the output sets are smoothed by the contribution of the noise "balls", and because we have assumed a unique $\overline{y}$, the separating hyperplane for the output sets is tangent to both sets, and at any given time it is unique. By inserting a known output from one of the models into the equation of the hyperplane (defined by its normal and a point on the plane, in this case $\overline{y}$), it can be determined whether that set lies above or below the plane. By inserting an output, the origin of which is unknown, into the equation of the hyperplane, and subsequently observing the sign of the result, we can determine from which model that output originates, and thus accomplish model identification.

Mathematically, the existence of the separating hyperplane implies that there is a function $a(t) \in L^2$ such that if we define

$$\phi(y) = \langle a, y - \overline{y} \rangle = \int_0^{t_f} a(t)^T [y(t) - \overline{y}(t)] \, dt \tag{3.3}$$

we have that ϕ is nonnegative on one $\mathcal{A}^i(v)$ and nonpositive on the other $\mathcal{A}^i(v)$. The function $a(t)$ is the normal to the separating hyperplane. We call ϕ the *test function.*

Numerical and system error can cause the output sets to overlap when $\widehat{v}$ is applied. To compensate for this effect one would likely apply $\delta\widehat{v}$ with δ a little more than one, resulting in output sets from the two models which are such a distance apart that their closures do not intersect. Thus, it will be important to have a test function that works for detection signals larger than $\widehat{v}$.

Recall from Section 2.1.1, the output equation for each model is

$$y = C_i\mathcal{L}_i(B_iv) + C_i\mathcal{L}_i(M_i\mu_i) + C_ie^{A_it}\xi_i + N_i\mu_i \tag{3.4}$$

where $C_i\mathcal{L}_i(B_iv)$ is a vector depending linearly on v. Applying δv translates $\mathcal{A}^i(0)$ by $\delta C_i\mathcal{L}_i(B_iv)$. Using $\delta > 1$ causes translation of $\mathcal{A}^i(v)$ by $(\delta - 1)C_i\mathcal{L}_iB_iv$. The vector $\overline{y}$ was on the boundary of $\mathcal{A}_i(\widehat{v})$. Thus $\overline{y} + (\delta - 1)C_i\mathcal{L}_iB_i\widehat{v}$ is now on the boundary of $\mathcal{A}_i(\delta\widehat{v})$ and the two output sets are disjoint. As a result, the supporting hyperplane at either $\overline{y} + (\delta - 1)C_0\mathcal{L}_0B_0\widehat{v}$ or $\overline{y} + (\delta - 1)C_1\mathcal{L}_1B_1\widehat{v}$ is still a separating hyperplane (not unique) and the normal is still $a(t)$. Therefore, we may define the test function

$$\phi_\delta(y) = \int_0^{t_f} a(t)^T(y(t) - \overline{y}(t) - (\delta - 1)q(t))dt \tag{3.5}$$

where $q(t) = \tau C_0\mathcal{L}_0B_0\widehat{v} + (1 - \tau)C_1\mathcal{L}_1B_1\widehat{v}$ for a fixed $0 \leq \tau \leq 1$. Note that choosing q this way with $0 < \tau < 1$ makes the test strictly positive on one output set and strictly negative on the other, which is more computationally robust. In the examples of Chapter 4 we choose $\tau = \frac{1}{2}$. See Figure 3.1 for a finite dimensional depiction of this discussion.

3.1.3 Approximating the Separating Hyperplane

Unfortunately, no analytic characterization exists for the boundaries of the output sets at $\overline{y}$. This characterization is required for direct computation of the tangent at the point of intersection. Thus, the equation of the separating (tangent) hyperplane is hard to compute. One way to compensate for the lack of a characterization of the output set boundaries at $\overline{y}$ is to artificially force the sets apart. Suppose we have done so, and let the points on the boundary of each set which are closest to each other be called $\overline{y}_0$ and $\overline{y}_1$, respectively. The equation of the line segment $(\overline{y}_0 - \overline{y}_1)$ will be the normal to a separating hyperplane for the forced-apart sets. This normal, along with a point on the line segment, can be used to define the separating hyperplane. If the

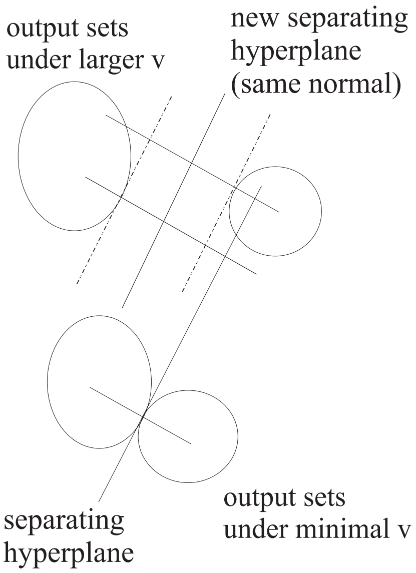

Figure 3.1: Output sets under application of $\widehat{v}$ and $\delta\widehat{v}$, $\delta > 1$

sets have been separated correctly, this separating hyperplane will approximate the separating hyperplane for the original output sets.

The accuracy of the approximation depends on how the sets are forced apart. One way to do it is to shrink the contribution of the noise vectors. Since the noise contribution to the output is a (possibly misshapen) "ball", multiplying the noise by a factor, $\epsilon < 1$, will force the output sets apart in such a way that the accuracy of the approximation can be made better by selection of a larger ϵ.

In fact, reducing the noise contribution is equivalent to increasing the detection signal. To see this fact, consider a generic model under the application of minimal energy detection signal, $\widehat{v}$, and reduced noise, $\epsilon\mu$.

$$x' = Ax + B\widehat{v} + M\epsilon\mu \tag{3.6a}$$

$$y = Cx + N\epsilon\mu. \tag{3.6b}$$

Multiplying both equations by $\frac{1}{\epsilon}$, we obtain

$$\begin{aligned} \left(\frac{x}{\epsilon}\right)' &= A\left(\frac{x}{\epsilon}\right) + B\left(\frac{\widehat{v}}{\epsilon}\right) + M\mu && (3.7a) \\ \left(\frac{y}{\epsilon}\right) &= C\left(\frac{x}{\epsilon}\right) + N\mu. && (3.7b) \end{aligned}$$

Letting $z = \frac{x}{\epsilon}$, $w = \frac{y}{\epsilon}$ and $\delta = \frac{1}{\epsilon}$, (3.7) becomes

$$\begin{aligned} z' &= Az + B\delta\widehat{v} + M\mu && (3.8a) \\ w &= Cz + N\mu && (3.8b) \end{aligned}$$

which is just the original generic model under the application of larger than minimal detection signal, $\delta\widehat{v}$, but with full noise contribution, μ. This equivalence is important in that it allows us to use the model identification test function (3.5) for the reduced noise problem as well as for the larger-than-minimum energy detection signal problem. In terms of the output sets, the equivalence may be stated as follows: if $\mathcal{A}^i_\epsilon(v)$ is the output from model i using detection signal v and noise weighting ϵ, so that $\mathcal{A}^i_1(v) = \mathcal{A}^i(v)$, then $\mathcal{A}^i_\epsilon(v) = \epsilon\mathcal{A}^i(\frac{1}{\epsilon}v)$.

3.1.4 Problem Statement

The discussion of the preceding section leads us to a new form for the system models.

$$\begin{aligned} x_i' &= A_i x_i + B_i\widehat{v} + M_i\epsilon\mu_i && (3.9a) \\ y_i &= C_i x_i + N_i\epsilon\mu_i && (3.9b) \end{aligned}$$

for $i = 0$ and 1. To find the separating hyperplane for the output sets, we must find the points on the boundaries of the closures of the sets that are closest to each other. In order to include the boundaries of the otherwise open output sets, we must change our bounded energy noise model to

$$\|\mu_i\|^2 \leq 1, \quad i = 0, 1. \tag{3.10}$$

To compute the normal to the separating hyperplane, simply minimize $\|y_0 - y_1\|^2$ subject to (3.9-3.10) using an optimal control package. SOCS is quite capable of handling this problem. The solutions $\overline{y}_0$ and $\overline{y}_1$ obtained can be differenced and normalized to compute the normal to the separating hyperplane. Any point on the line segment between $\overline{y}_0$ and $\overline{y}_1$ may be used as the defining point on the hyperplane. See Figure 3.2 for a finite dimensional depiction of this discussion.

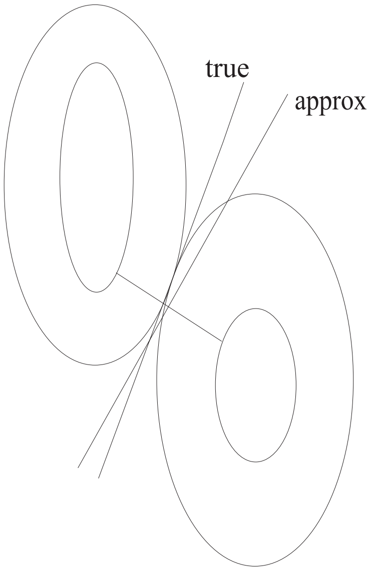

Figure 3.2: Output sets under full and reduced noise contributions

As stated earlier, the accuracy of the approximation obtained from this problem can be improved simply by selecting a larger ϵ. A bound on the error is characterized in the next section. With the problem now fully defined, we present the model identification algorithm.

3.2 The Model Identification Algorithm

As described in Chapter 2, Boeing's SOCS package [3, 4] efficiently solves the optimization problem described above. MATLAB, by The MathWorks, Inc. [26] is convenient for subsequent computations. While we do not use MAPLE, by Waterloo Maple, Inc. [18] to generate FORTRAN code as for the MEDS algorithm, higher dimensional problems will make it desirable to do so. The model identification (MI) algorithm (with appropriate software in parentheses):

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

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

3. Perform constrained optimization (SOCS)

$$\min \|y_0 - y_1\|^2 \tag{3.11a}$$

   subject to the constraints

$$\begin{aligned} x_i' &= A_i x_i + B_i \widehat{v} + M_i \epsilon \mu_i && (3.11\text{b}) \\ y_i &= C_i x_i + N_i \epsilon \mu_i && (3.11\text{c}) \\ q_i' &= \mu_i^T \mu_i, \;\; q_i(0) = 0, \;\; q_i(t_f) \le 1 && (3.11\text{d}) \end{aligned}$$

   for $i = 0$ and 1

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

5. Compute $a_\epsilon(t)$, the normal to the separating hyperplane (MATLAB)

$$a_\epsilon = \frac{\overline{y}_{0,\epsilon} - \overline{y}_{1,\epsilon}}{\|\overline{y}_{0,\epsilon} - \overline{y}_{1,\epsilon}\|} \tag{3.12}$$

6. Compute $\overline{y}_\epsilon(t)$, the point on the separating hyperplane, as the midpoint of the line segment connecting $\overline{y}_{0,\epsilon}$ and $\overline{y}_{1,\epsilon}$ (i.e., $\tau = \frac{1}{2}$) (MATLAB)

$$\overline{y}_\epsilon = \frac{\overline{y}_{0,\epsilon} + \overline{y}_{1,\epsilon}}{2} \tag{3.13}$$

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

$$\phi_\epsilon(y_0) = \langle a_\epsilon, y_0 - \overline{y}_\epsilon \rangle \quad > 0$$

$$\phi_\epsilon(y_1) = \langle a_\epsilon, y_1 - \overline{y}_\epsilon \rangle \quad < 0$$

   or vice versa, where y_i is an unknown output from model i, $i = 0$ or 1

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

Note that as $\epsilon \to 1$, the computed normal approaches the true normal. Unfortunately, numerical error in (3.12) increases when $(1 - \epsilon)$ is very small due to division by small numbers. Thus, $(1 - \epsilon)$ should be chosen sufficiently large to ensure a discrete distance between output sets, in order to reduce the effect of numerical error. However, $(1 - \epsilon)$ must be small enough to ensure that the skewing effect from the weight matrices on the noise inputs are included, in order to increase the accuracy of the computed normal.

The following theorem summarizes the accuracy of the computed test function [12].

THEOREM 3.1. *Let $a_\epsilon(t)$ be the normal from the MI algorithm using $0 < \epsilon < 1$ and let $\overline{y}_{0,\epsilon}, \overline{y}_{1,\epsilon}$ be the values of y_0, y_1 that give the minimum distance. Let $\overline{y}_\epsilon =$*

$\frac{1}{2}(\overline{y}_{0,\epsilon} + \overline{y}_{1,\epsilon})$. *Let the hyperplane test be*

$$\phi_\epsilon(w) =< a_\epsilon(t), w - \overline{y}_\epsilon > . \tag{3.14}$$

Let $\delta(\epsilon) = \frac{\|\overline{y}_{0,\epsilon} - \overline{y}_{1,\epsilon}\|}{2}$. *Then there is a constant* K *so that*

$$\delta(\epsilon) \leq K(1-\epsilon) \tag{3.15}$$

and if $w \in (\mathcal{A}^0_\epsilon(\widehat{v}) \cup \mathcal{A}^1_\epsilon(\widehat{v}))$*, then*

$$\phi_\epsilon(w) \geq \delta(\epsilon) \implies w \in \mathcal{A}^0_\epsilon(\widehat{v}) \tag{3.16}$$

$$\phi_\epsilon(w) \leq -\delta(\epsilon) \implies w \in \mathcal{A}^1_\epsilon(\widehat{v}). \tag{3.17}$$

Proof (from [12]) (3.16) and (3.17) follow from noting that $\phi_\epsilon(w) = \delta(\epsilon)$ is the supporting hyperplane of $\mathcal{A}^0_\epsilon(\widehat{v})$ at $\overline{y}_{0,\epsilon}$ while $\phi_\epsilon(w) = -\delta(\epsilon)$ is a parallel supporting hyperplane of $\mathcal{A}^1_\epsilon(\widehat{v})$ at $\overline{y}_{1,\epsilon}$. K can be taken as $\|C_0\mathcal{L}_0M_0 + N_0\| + \|C_1\mathcal{L}_1M_1 + N_1\|$. □

K is related to the linear operators applied to the noise vectors and is thus a measure of the amount of skewing that occurs in the output sets due to the noise input. In practice, the tests are often much better than the theorem indicates but the result allows for highly skewed convex sets.

It is important to note that the MI algorithm can be used if the applied detection signal is larger than the minimal proper. If v is proper and not minimal, then one can set $\epsilon = 1$ in the algorithm and still obtain a nonzero distance between the output sets, due to the equivalence between the reduced noise and increased detection signal approaches. In fact, one can use a combination of both reduced noise input and increased detection signal, which is useful when it is not known whether the detection signal to be applied is minimal or not. This will be important when we discuss the case with multiple fault models. A sample driver file for the MI algorithm coded in

SOCS, along with a MATLAB m-file for subsequent computations are in Appendix A.

3.3 Variations

Like the MEDS algorithm, applications of the MI algorithm are limited by the assumptions and formulation of the problem. Fortunately, the assumptions and formulation of the model identification problem are much more conducive to extensions than the minimum energy detection signal problem. Extensions are easily made to multiple fault models, alternative formulations, pre-existing controls, alternative cost functions, and different initial conditions.

3.3.1 Multiple fault models

The result of the MEDS algorithm consists of a group of disjoint output sets, one for each possible model. Since we have no *a priori* knowledge of the spatial locations of these sets, the MI algorithm must be used to separate each pair of sets. If there are n possible models, then there will be $\frac{n(n-1)}{2}$ separating hyperplanes. Each output set will exhibit a unique combination of test function signs (positive/negative), for those test functions which can discriminate that output set.

For example, take the case in which three models exist for the system, and let the output sets of the models be called Y_0, Y_1, and Y_2. Let the hyperplane separating Y_0 and Y_1 be called H_{01}, separating Y_0 and Y_2 be called H_{02}, and separating Y_1 and Y_2 be called H_{12}. Finally, let the signs of the test function for output y be $(sgn(H_{01}), sgn(H_{02}), sgn(H_{12}))_y$. Suppose the known output from Y_0 exhibits $(+, -, *)$, from Y_1 exhibits $(-, *, +)$, and from Y_2 exhibits $(*, +, -)$, where $*$ indicates the inability of that test function to discriminate the output set. Then, a valid unknown output exhibiting any possible combination of test function signs will fall into one and only one of the output sets. Note that it is not possible for certain combinations of test function signs to be exhibited by a valid output, $(+, +, +)$, for instance.

It should be noted that the minimum energy detection signal for the multiple fault model problem may not be minimal for any of the pairwise problems. It will always be proper due to the construction of the combined problem, but it may be larger than required. In addition, it may not be apparent for which pairwise problem the detection signal is larger than minimal. While approaches such as that of [30] must use the minimal energy detection signal, the algorithm and test function described in this chapter can still be applied when the detection signal may not be minimal energy.

Finally, if the goal is to detect a fault, but it is not important to distinguish between fault models, the detection signal should be minimized by solving only those pairwise problems including the nominal model (i.e., in the three model case, model 0 versus model 1, and model 0 versus model 2). The nominal model will be distinguishable from the fault models, but more than one separating hyperplane will be required. Outputs from the fault models may wrap around the nominal model, in which case a single hyperplane will not have the nominal model on one side and all fault models on the other side. Examples of the multiple fault model problem are included in the next chapter.

3.3.2 Alternative Formulations

Since the MI algorithm formulates the problem in terms of the original system matrices, and no reduction in system dimension is attempted, the problem is already in unreduced form. Thus, since parameters representing physical quantities are not combined and sparsity is not lost, the question of whether or not to reduce the model is not an issue as it was in the MEDS algorithm. While alternative formulations are probably available, none could describe the problem as plainly, and in such a straightforward manner as the present formulation.

3.3.3 Controlled Systems

As was the case for the MEDS algorithm, the presence of a known control does not present any difficulty for the MI algorithm. The output of the MEDS algorithm combines the known control with the detection signal. This combined signal is merely a known input to the MI algorithm, and the fact that a control is present is transparent to the algorithm. Other types of controls as described in Chapter 2 are equally as transparent.

3.3.4 Alternative Cost Functions

The issue of alternative cost functions arose in the MEDS algorithm because viable alternative costs exist for that problem. In the MI algorithm, the cost function is merely a means by which to compute the separating hyperplane via the two closest points on the closure of each output set. Thus it is the closest points which are of interest, and not the optimal cost. Therefore, the issue of alternative cost functions is not important at all to the model identification problem.

3.3.5 Knowledge of Initial Conditions

The variation of the noise model presented in the “Variations” section of Chapter 2 is as transparent to our MI algorithm as it is to the MEDS algorithm. The only change occurs in the boundary conditions for the differential equation.

Recall, however that variations on the initial conditions led Nikoukhah, et al. [30] in a different direction for computing $\widehat{v}$. While the separating hyperplane approach is also utilized in [30], the method for computing it is quite different. Instead of formulating an optimization problem, one of the Riccati equations solved while computing $\widehat{v}$ is used along with a new BVP. The normal to the separating hyperplane is shown to be related to the Lagrange multiplier of the detection signal Riccati equation. Care must be taken to avoid introducing large errors into the solution via the integration, but otherwise the computation of the normal is a simple BVP calculation. The shortcoming of the approach is that it must use $\widehat{v}$, and not a larger multiple. By its nature, the calculation uses the detection signal Riccati equation that produces $\widehat{v}$, so $\delta\widehat{v}$, where $\delta > 1$, is not available. The MI algorithm has already been shown to be robust to the use of larger multiples of $\widehat{v}$.

3.3.6 Conclusion

It is clear from the problem formulation, the algorithm description, and the above exploration of possible variations to the model identification problem that it is much simpler and more straight forward than the minimum energy detection signal problem. The question of flexibility, i.e., how easy it is to adapt the algorithm to a larger set of problems, is therefore quite easily answered: the MI algorithm can be adapted to handle any problem the MEDS algorithm can handle. As was stated in the previous chapter, while current theory for the multiple fault model case only provides for problem development, it does not provide a practical means with which to solve the problem. Our MEDS and MI algorithms not only provide theoretical development for the multiple fault model case, they also supply a simple method for solving such problems.

Chapter 4

Examples and Analysis of Results

4.1 The Complete Problem and Algorithm

Before beginning a discussion of the various examples to which the MEDS and MI algorithms have been applied, it is beneficial to review the complete problem and combined algorithm. Recall from Chapter 2, that the true model of the system is one of two models

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

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

for $i = 0$ and 1. Our first goal is to apply the minimum energy detection signal, v, such that the (convex) output sets of the two models are disjoint. Thus, a given output may be derived from only one model. Our second goal is to compute the equation of the hyperplane which separates the two output sets. From this equation we define the test function. The substitution of known outputs from each model into the test function will indicate the sign ($+/-$) that each set will exhibit. The substitution of an output of unknown origin into the test function will result in a positive number if it is derived from one model and a negative number if it is derived from the other model. Thus the correct model for the system will be identified.

These two goals are accomplished by the fault detection and model identification (FDMI) algorithm (the combination of the MEDS and the MI algorithms), repeated below (note: only the Version Two, non-Riccati form is repeated here, see Chapter 2 for other developed forms)

1. Perform QR decomposition on N_i^T

$$N_i^T = Q_i R_i \tag{4.2}$$

where the Q_i are unitary matrices. Thus

$$N_i = R_i^T Q_i^T \tag{4.3}$$

2. Perform constant orthogonal coordinate changes on μ_i

   (a) Let $R_i^T = [\overline{N}_i \;\; 0]$, where $\overline{N}_i$ is invertible

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

$$N_i \mu_i = [\overline{N}_i \;\; 0] \begin{pmatrix} \overline{\mu}_i \\ \widetilde{\mu}_i \end{pmatrix} \tag{4.4}$$

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

$$M_i \mu_i = M_i Q_i^{-T} Q_i^T \mu_i = [\overline{M}_i \;\; \widetilde{M}_i] \begin{pmatrix} \overline{\mu}_i \\ \widetilde{\mu}_i \end{pmatrix} \tag{4.5}$$

   Note the system model, (4.1), becomes

$$x_i' = A_i x_i + B_i v + \overline{M}_i \overline{\mu}_i + \widetilde{M}_i \widetilde{\mu}_i \tag{4.6a}$$

$$y = C_i x_i + \overline{N}_i \overline{\mu}_i \tag{4.6b}$$

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

(a) Combine both equations (4.6b) for y, solve for $\overline{\mu}_0$, and substitute into (4.6a), $i = 0$

$$x_0' = (A_0 - \overline{M}_0\overline{N}_0^{-1}C_0)x_0 + \overline{M}_0\overline{N}_0^{-1}C_1x_1 + B_0v + \widetilde{M}_0\widetilde{\mu}_0 + \overline{M}_0\overline{N}_0^{-1}\overline{N}_1\overline{\mu}_1 \tag{4.7}$$

(b) Let

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

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

Note the system model, (4.6), is now

$$x' = Ax + Bv + M\mu \tag{4.8}$$

4. Compute new system matrices

(a) Let $C = [\overline{N}_0^{-1}C_0 \quad -\overline{N}_0^{-1}C_1]$ and $N = \overline{N}_0^{-1}[0 \quad \overline{N}_1 \quad 0]$, with the columns of N conforming to μ

(b) Let $H = -4\beta C^T N$ and $Q = 2\beta C^T C$

(c) Let

$$R = 2\begin{bmatrix} \beta I & 0 & 0 \\ 0 & (1-\beta)I + \beta\overline{N}_1^T\overline{N}_0^{-T}\overline{N}_0^{-1}\overline{N}_1 & 0 \\ 0 & 0 & (1-\beta)I \end{bmatrix}$$

with its rows and columns conforming to μ

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

5. Perform constrained optimization. Solve

$$(\gamma^*)^{-2} = \min_{v,\beta} \int_0^{t_f} \|v\|^2 dt \tag{4.9a}$$

subject to the constraints

$$\begin{aligned} x' &= Ax + Bv + M\mu & (4.9b)\\ \lambda' &= -Qx - \frac{1}{2}H\mu - A^T\lambda, \quad \lambda(0) = \lambda(t_f) = 0 & (4.9c)\\ Z' &= \frac{1}{2}\left[x^TQx + x^TH\mu + \mu^TR\mu\right], \quad Z(0) = 0, \quad Z(t_f) \geq 1 & (4.9d)\\ 0 &= R\mu + \frac{1}{2}H^Tx + M^T\lambda & (4.9e)\\ & 0.01 \leq \beta \leq 0.99 & (4.9f) \end{aligned}$$

6. Let $\widehat{v}$ be the minimum energy proper detection signal

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

8. Perform constrained optimization. Solve

$$\min \|y_0 - y_1\|^2 \tag{4.10a}$$

subject to the constraints

$$\begin{aligned} x_i' &= A_ix_i + B_i\widehat{v} + M_i\epsilon\mu_i & (4.10b)\\ y_i &= C_ix_i + N_i\epsilon\mu_i & (4.10c)\\ q_i' &= \mu_i^T\mu_i, \quad q_i(0) = 0, \quad q_i(t_f) \leq 1 & (4.10d) \end{aligned}$$

for $i = 0$ and 1

9. Let $\overline{y}_{0,\epsilon}$ and $\overline{y}_{1,\epsilon}$ be the closest points computed by the optimization

10. Compute $a_\epsilon(t)$, the normal to the separating hyperplane

$$a_\epsilon = \frac{\overline{y}_{0,\epsilon} - \overline{y}_{1,\epsilon}}{\|\overline{y}_{0,\epsilon} - \overline{y}_{1,\epsilon}\|} \tag{4.11}$$

11. Compute $\overline{y}_\epsilon(t)$, the point on the separating hyperplane, as the midpoint of the line segment connecting $\overline{y}_{0,\epsilon}$ and $\overline{y}_{1,\epsilon}$ (i.e., $\tau = \frac{1}{2}$)

$$\overline{y}_\epsilon = \frac{\overline{y}_{0,\epsilon} + \overline{y}_{1,\epsilon}}{2} \tag{4.12}$$

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

$$\phi_\epsilon(y_0) = \langle a_\epsilon, y_0 - \overline{y}_\epsilon \rangle \quad > 0$$

$$\phi_\epsilon(y_1) = \langle a_\epsilon, y_1 - \overline{y}_\epsilon \rangle \quad < 0$$

or vice versa, where y_i is an unknown output from model i, $i = 0$ or 1

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

4.2 Introduction of Software

As alluded to in previous chapters, Boeing's Sparse Optimal Control Software (SOCS) is the main software in which the optimal control problems of the FDMI algorithm have been coded for this thesis. System matrices are first reduced using MATLAB. The reduced system matrices from MATLAB are inserted into the appropriate FORTRAN subroutines. The dimensions of the reduced model's matrices are fed into

MAPLE for the construction of the constraint equations. MAPLE's FORTRAN translator is used to convert these equations into FORTRAN code which are then pasted into FORTRAN subroutines for compilation and execution. MAPLE is used to compute the constraint equations symbolically, so that the problems may by kept in parameterized form. MATLAB is used to analyze and plot the output from the SOCS routines

4.2.1 SOCS Parameters

In addition to the specific parameters for a given problem, SOCS allows the user to set various general parameters which control the optimization method and convergence tolerances. In Chapter 1, we described several discretization methods implemented by SOCS. By setting two control parameters we have selected the Compressed Hermite-Simpson discretization method for all examples. While it is possible to substitute another method to solve the resulting finite dimensional problem, we have not done so, but instead have used the default SQP method supplied in SOCS. This combination of discretization method and nonlinear program solver exhibits good convergence properties and efficient use of central processor unit (CPU) time and is well suited to the types of differential equations, and L^2 and algebraic constraints present in our problem set. Options that control the operation and convergence of the optimization program are:

- Sparsity - If set to "sparse," the program takes advantage of sparsity in user supplied constraints. Our constraints are sparse, so this option is used.
- Initial guess type - The user has several options available to supply an initial guess for the optimization program. Choices range from a linear interpolation between endpoint values, to construction of an initial guess

from a user-supplied B-spline definition or explicit calculation at each of the initial mesh points. All of our examples use the linear interpolation option. While this often results in an infeasible initial guess, in every case the program rectifies this condition after a few constraint satisfaction iterations.

- Initial mesh size - The user sets the size of the initial mesh. To reduce unnecessary computation, the initial mesh should be the coarsest possible that allows the program to attain constraint satisfaction after a few iterations. If the initial guess is already feasible, the initial mesh size should be left at the default value of 10. Several of our examples require a finer initial mesh than the default in order to attain constraint satisfaction.
- Discretization order and stage number - When set to the appropriate values, the program uses a high order discretization method and stage number for mesh refinement. While not used in our examples, this option may be useful in higher dimensional problems.
- Tolerances - The user may set tolerances for the relative error in the objective function and the differential equation constraints, as well as various absolute errors. These tolerances may be set to different values for the two different phases of the optimization (discretization and nonlinear program solver). Several of our tolerances are set tighter than default values to obtain more accurate solutions, resulting in modest increases in CPU times.

Samples and discussions of the MATLAB, MAPLE, and SOCS routines used to generate the results of this chapter are depicted in Appendix A.

4.2.2 Choosing a Value of ϵ

In the previous chapter, we mentioned that difficulty may arise in choosing the value of the noise multiplier, ϵ, in the model identification part of the algorithm. There we noted that as $\epsilon \to 1$, the computed normal to the separating hyperplane approaches the true normal, but numerical error increases also. We concluded that $(1-\epsilon)$ should be chosen sufficiently large to ensure a discrete distance between output sets, in order to reduce the effect of numerical error. However, $(1-\epsilon)$ must be small enough to ensure that the skewing effect from the weight matrices on the noise inputs are included, in order to increase the accuracy of the computed normal.

To test the effect of different values of ϵ on the accuracy of the computed normal, we ran several different models with different values of ϵ. We found that the normal is not very sensitive to the value of ϵ as long as ϵ is not too small. To ensure a discrete distance between output sets without reducing the skewing effect of the coefficient matrices on the noise vectors, we chose $\epsilon = 0.7$ for all examples. A sample plot of the computed normal of one example problem for $\epsilon = 0.3, 0.5, 0.7, 0.9$ is shown in Figure 4.1.

4.3 Introduction of Examples

To test the operation of the algorithm, as well as to examine how results vary with the problem, we have applied the FDMI method to fourteen examples. Four of these examples are one-dimensional problems, and are examined to shed light on the performance of the algorithm on stable-to-stable and stable-to-unstable fault situations. One of these examples is coded in all four formulations of the algorithm in order to compare the relative computational performance of the different formulations. Seven examples are two-dimensional problems, and are examined to shed light on several

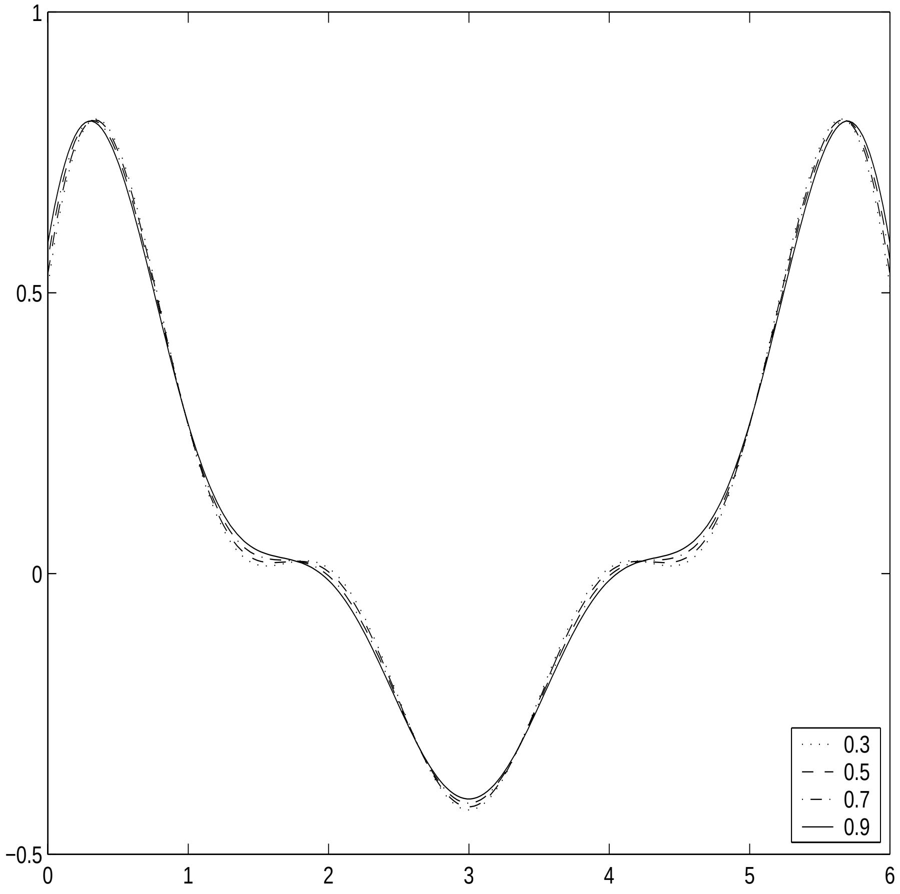

Figure 4.1: Typical variation of $a_\epsilon(t)$ with ϵ

phenomena of interest. These phenomena include:

- $\widehat{v}$, γ^*, $\overline{y}_\epsilon$ and $a_\epsilon(t)$, and how each varies with the problem and the interval length,
- how results differ between stable-to-stable and stable-to-unstable oscillatory systems (i.e., with imaginary or complex eigenmodes),
- whether the algorithm will work for systems with common modes (i.e., unobservable systems).

Another example is three-dimensional, a real world control system; it is examined to determine the efficiency of the algorithm for larger scale systems. The last two examples are each systems with a nominal model and two fault models. One of these is one-dimensional and the other is two-dimensional. These examples are included to demonstrate how the algorithm handles multiple fault model systems. Overall, the example suite provides the basis for a comprehensive analysis of the FDMI algorithm, its strengths, as well as a few of its weaknesses.

4.4 One-Dimensional State Examples

To begin our analysis, we look at four one-dimensional examples. The very first example is coded in all four formulations of the algorithm. This is done both to confirm results between formulations and to attempt to determine if one formulation is more efficient than the others. The remaining examples shed light on the shape and energy of the minimum energy detection signal, $\widehat{v}$, as well as the shapes of the normal to the separating hyperplane, $a_\epsilon(t)$, the midpoint of the shortest line segment between the output sets, $\overline{y}_\epsilon(t)$, and the separability index, γ^*, for one-dimensional problems.

4.4.1 Primary One-Dimensional Example

We begin with the simplest problem for which all terms appear.

EXAMPLE 4.1. *[Change of eigenvalue, stable-to-stable] This problem corresponds to the case where there has been an internal change in some system parameter such as friction in a joint or resistance of a resistor.*

$$x_0' = -2x_0 + v + \mu_2 \tag{4.13a}$$

$$y = x_0 + \mu_1 \tag{4.13b}$$

$$x_1' = -x_1 + v + \mu_4 \tag{4.13c}$$

$$y = x_1 + \mu_3. \tag{4.13d}$$

This problem, as well as each subsequent example, is formulated such that the constant change of coordinates described in the algorithm is unnecessary. Model reduction is much more straightforward with this formulation, but no generality is lost in the process. Figures 4.2-4.3 depict $\widehat{v}$ for $t_f = 1, 10, 20, 100$ respectively. Figure 4.4 shows $\widehat{v}$ for $t_f = 20, 100$ plotted to the same scale.

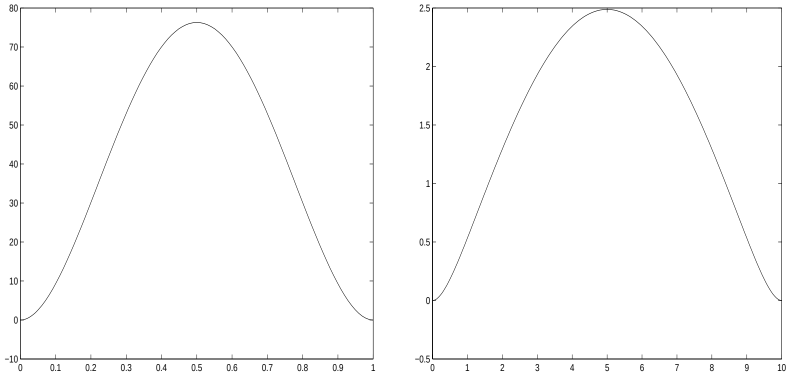

Figure 4.2: $\widehat{v}$, for Example 4.1: $t_f = 1$ (left), $t_f = 10$ (right)

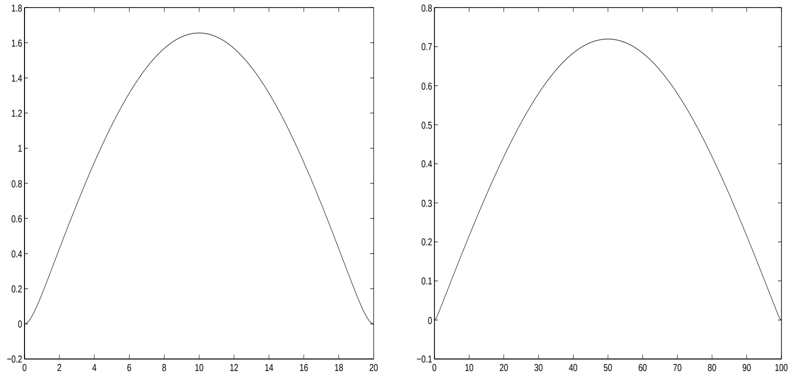

Figure 4.3: $\widehat{v}$ for Example 4.1: $t_f = 20$ (left), $t_f = 100$ (right)

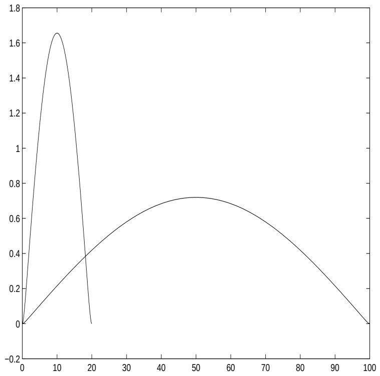

Figure 4.4: $\widehat{v}$ for Example 4.1: $t_f = 20, 100$

Figure 4.5 shows the relationship between γ^* and the interval length. Table 4.1 compares γ^* and $\widehat{v}$ for various interval lengths.

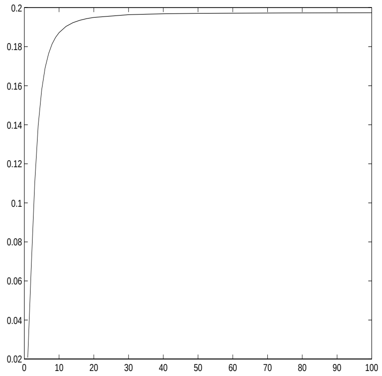

Figure 4.5: γ^* for Example 4.1 as a function of t_f

Table 4.1: γ^* and $||\widehat{v}||$ for Example 4.1: $t_f = 1, 10, 20, 100$

t_f	γ^*	$\|\|\widehat{v}\|\|$
1	0.02075	48.1919
10	0.18719	5.3423
20	0.19498	5.1288
100	0.19736	5.0669

As the plots and the table show, the energy of $\widehat{v}$ decreases as the interval length increases. However, once a threshold interval length is reached, no significant decrease in the energy of $\widehat{v}$, or increase in γ^* occurs. It should be noted that as the interval length decreases, γ^* becomes very small, indicating the difficulty involved in distinguishing between the two models on very short intervals. Since $||\widehat{v}||$ is equal to the reciprocal of γ^*, the energy of the detection signal will be extremely large on very short intervals. The threshold for Example 4.1 appears to be near $t_f = 15$, so that in the case where $t_f = 1$ the energy of $\widehat{v}$ is quite significant.

As to the shape of $\widehat{v}$ on different intervals, Figure 4.6 compares $\widehat{v}$ for the different intervals after rescaling in both magnitude and duration. The innermost line is $\widehat{v}$ for $t_f = 1$, continuing outward so that the outermost line is $\widehat{v}$ for $t_f = 100$. It is clear that $\widehat{v}$ is not the same function on different intervals. In [29] it was observed that $\widehat{v}$ for the discrete problem begins to resemble the product of sine functions as the interval lengthens. Figure 4.6 also demonstrates this resemblance, though in a more approximate sense than was demonstrated in [29].

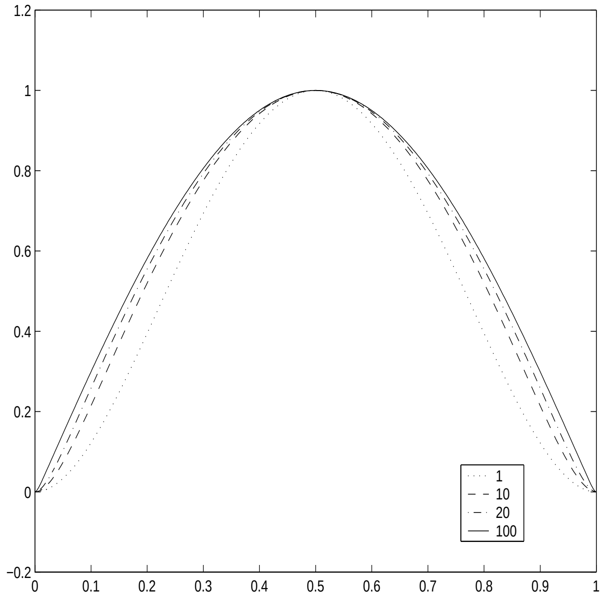

Figure 4.6: Rescaled $\widehat{v}$ for Example 4.1: $t_f = 1, 10, 20, 100$

Figure 4.7 gives the $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for various values of ϵ. For Example 4.1, the noise input is not skewed at all by its coefficient matrix, so we would expect the separating hyperplane to be insensitive to the value of ϵ. Figure 4.7 clearly shows that $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ do not vary significantly across the full spectrum of possible values of ϵ for this problem.

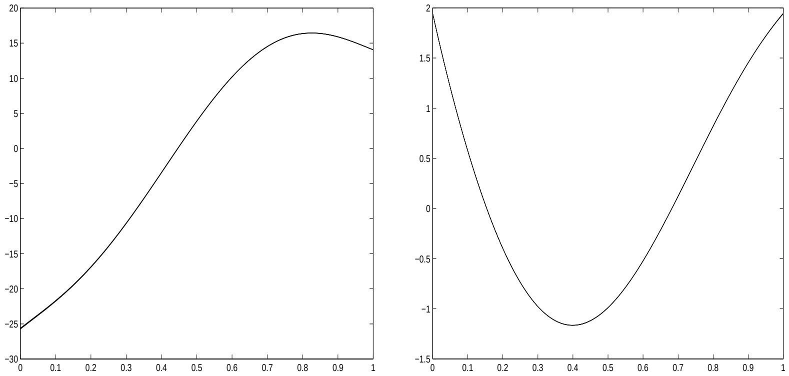

Figure 4.7: $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.1: $\epsilon = 0.3, 0.5, 0.7, 0.9$

As mentioned previously, Example 4.1 is coded in all four formulations of the FDMI algorithm. The results discussed above are from the Version One, non-Riccati form of the algorithm, but thorough comparisons have shown that the results are equivalent from all four forms. Table 4.2 summarizes the performance of all four formulations: Version One, non-Riccati (1nR) and Riccati (1R), and Version Two, non-Riccati (2nR) and Riccati (2R). Each was run on the intervals $[0, 1]$, $[0, 10]$, and $[0, 20]$ for a total of twelve runs. Ten of the runs use an initial mesh size of 50. For $t_f = 1$, the 2R formulation would not converge using this initial mesh size, as was also the case for $t_f = 20$ in the 1nR formulation. Both converged using an initial mesh size of 11, so that value was substituted. Those entries in the table are in boldface type. Non-convergence of the SOCS program occurs occasionally and is usually due to the combination of infeasible initial conditions and a failure to resolve system dynamics on a specific mesh. In addition, a few of the problems produced early program terminations due to the magnitude of the objective function. With an objective function on the order of 10^{-4}, the change in objective function values between iterations is commonly on the order of 10^{-6}. The default SOCS objective function error tolerance is 10^{-5}, and thus productive reductions in the objective function may be lost due to program termination before they occur. To counteract this effect, the objective function for those problems exhibiting this phenomenum is rescaled by 10^6. The scaling factor was removed before the data was tabulated. Later, we refer to this rescaling of the objective function as *conditioning.*

Table 4.2: Formulation comparison of Example 4.1: $t_f = 1, 10, 20$

	$t_f = 1$		$t_f = 10$		$t_f = 20$	
Form	CPU Time	Iterations	CPU Time	Iterations	CPU Time	Iterations
1nR	61.55	3	96.16	3	**67.04**	**6**
2nR	181.56	4	67.54	3	290.94	4
1R	72.89	4	68.6	3	130.14	4
2R	**138.17**	**6**	67.9	3	125.34	4

As the table indicates, results were not consistently better for any formulation. While formulation 1nR is as good as, or better than the others in four of the six columns, it is worse by 50% in the other two columns. Since there is no clear winner, and since it is also not our intent to recommend one formulation over any other, we chose to run the remaining models in formulations 1nR or 2nR. These formulations provide a more direct correlation with the original problem parameters and are simpler to encode. In addition, neither form requires Riccati equations. The direct transcription approach taken by SOCS converts every problem into a boundary value problem, even those in Riccati form. This translation removes all of the usual advantages of the Riccati approach. Formulation 2nR provides excellent compatibility between the minimum energy detection signal and model identification parts of the algorithm, while some rescaling of results is required in formulation 1nR. For this reason, our higher dimensional examples are coded only in formulation 2nR. The formulation used is transparent to the results presented below.

4.4.2 Other One-Dimensional Examples

The remaining one-dimensional problems are similar to Example 4.1. Plots of $\widehat{v}$ for each problem, as well as comparisons to Example 4.1 follow the last problem definition.

Example 4.2. *[Change of eigenvalue, stable-to-unstable] This is also a simple problem corresponding to a change in a system parameter, but here the fault model is unstable.*

$$x_0' = -x_0 + v + \mu_2 \tag{4.14a}$$

$$y = x_0 + \mu_1 \tag{4.14b}$$

$$x_1' = \frac{1}{2}x_1 + v + \mu_4 \tag{4.14c}$$

$$y = x_1 + \mu_3. \tag{4.14d}$$

Example 4.3. *[Severe change of eigenvalue, stable-to-stable] This problem is similar to Example 4.1, but the parameter change is more severe, indicating stiff dynamics.*

$$x_0' = -x_0 + v + \mu_2 \tag{4.15a}$$

$$y = x_0 + \mu_1 \tag{4.15b}$$

$$x_1' = -30x_1 + v + \mu_4 \tag{4.15c}$$

$$y = x_1 + \mu_3. \tag{4.15d}$$

Example 4.4. *[Severe change of eigenvalue, stable-to-unstable] This problem is similar to Example 4.2, but the unstable mode's parameter change is severe, indicating* *highly unstable dynamics.*

$$x_0' = -x_0 + v + \mu_2 \tag{4.16a}$$

$$y = x_0 + \mu_1 \tag{4.16b}$$

$$x_1' = 30x_1 + v + \mu_4 \tag{4.16c}$$

$$y = x_1 + \mu_3. \tag{4.16d}$$

Figure 4.8 gives $\widehat{v}$ for Examples 4.2 -4.4 for $t_f = 1$. Table 4.3 compares the separability indices and the energy of $\widehat{v}$ for Examples 4.1-4.4 on the intervals $[0, 1]$ and $[0, 10]$. As the plots show, the shape of $\widehat{v}$ is similar for all of the one-dimensional problems in the set. Example 4.2 admits a lower energy minimum proper detection signal than Example 4.1 on the same interval. This is as expected since the two models of Example 4.2 have more contrasting dynamics than in Example 4.1. Examples 4.3 and 4.4 exhibit even sharper changes in dynamics, and the much lower energy $\widehat{v}$ for each of those problems on the shorter interval bears witness to that fact.

As the interval lengthens, however, the differences narrow. On $[0, 10]$, Example 4.1 still requires significantly more energy in $\widehat{v}$ than the other examples. The unstable mode of Example 4.2 and the large changes in dynamics of Examples 4.3-4.4 still allow for easier separation of the models of these problems. The differences between the last three examples, however, are much smaller than on the $[0, 1]$ interval. In fact, Example 4.2 now requires less energy than the other two examples. This indicates the expected result: problems with more distinct models are not as sensitive to the test period length as those with similar models. Obviously, extremely short intervals will present difficulties for all problems, but problems with distinct models will be less sensitive on those intervals, i.e., they will produce a larger γ^* and will thus be easier to separate than problems with more similar models.

Figure 4.9 shows all of the $\widehat{v}$ curves on the same plot, each rescaled to a maximum

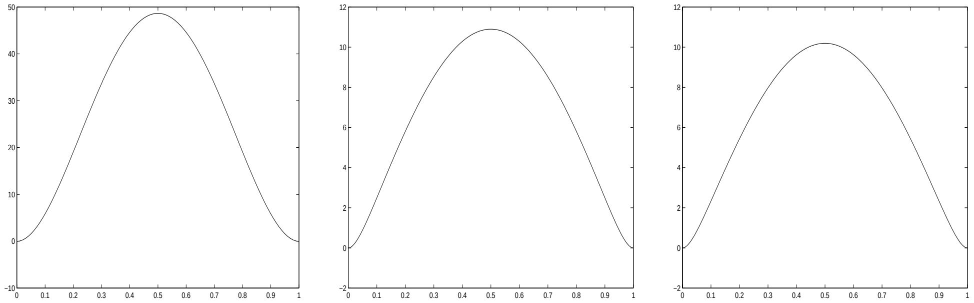

Figure 4.8: $\widehat{v}$ for Examples 4.2 (left), 4.3 (center), 4.4 (right): $t_f = 1$

Table 4.3: γ^* and $\|\widehat{v}\|$ for Examples 4.1-4.4: $t_f = 1, 10$

	$t_f = 1$		$t_f = 10$	
Example	γ^*	$\|\widehat{v}\|$	γ^*	$\|\widehat{v}\|$
4.1	0.02075	48.1919	0.18719	5.3423
4.2	0.03261	30.6690	0.69849	1.4317
4.3	0.13418	7.4524	0.38687	2.5848
4.4	0.14344	6.9717	0.41355	2.4181

height of one. It is interesting to look at this combined $\widehat{v}$ plot. With $t_f = 1$ the $\widehat{v}$ of Examples 4.1 and 4.2 are superimposed on each other, as are the $\widehat{v}$ of Examples 4.3 and 4.4. A mathematical examination reveals a maximum difference of about 0.003 between the $\widehat{v}$ of Examples 4.1 and 4.2. On a longer time interval, the two $\widehat{v}$ are still very similar but there is a more visible difference. The fact that the two $\widehat{v}$ are so similar suggests that for some classes of problems one could use the same $\widehat{v}$ for a number of different fault models, changing only the gain to ensure that it is proper.

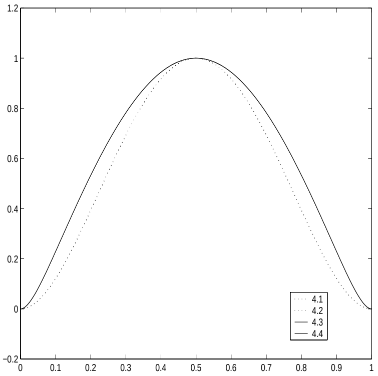

Figure 4.9: $\widehat{v}$ for Examples 4.1-4.4: $t_f = 1$

Figure 4.10 gives $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.4 for various values of ϵ. As we saw in previous plots of the parameters of the separating hyperplane, they are quite insensitive to the value of ϵ. In fact, for this problem, the plots for the different values of ϵ are superimposed on each other. The insensitivity of the parameters to the value of ϵ is encouraging in that it allows some assurance of computing a valid test function for virtually every possible output from competing models of the problem.

It should be noted that each $a_\epsilon(t)$ computed by the algorithm has been checked against those functions to which it should be orthogonal. That is, for the output

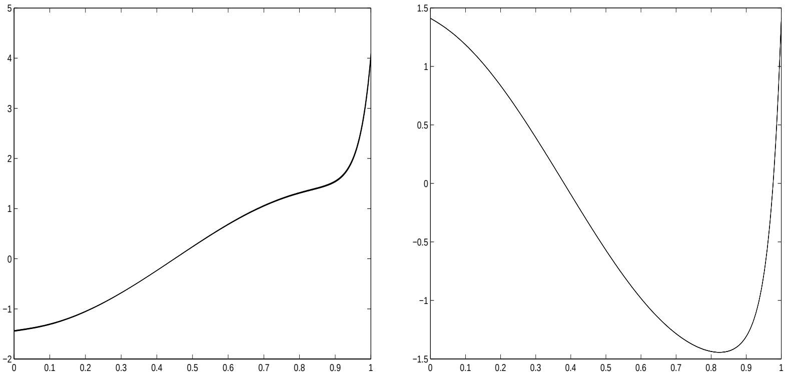

Figure 4.10: $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.4: $\epsilon = 0.3, 0.5, 0.7, 0.9$

given by

$$y = C_i\mathcal{L}_i(B_i\widehat{v}) + C_i\mathcal{L}_i(M_i\mu_i) + C_ie^{A_it}\xi_i + N_i\mu_i \tag{4.17}$$

the normal to the separating hyperplane should be orthogonal to $C_ie^{A_it}\xi_i$. The inner products of the computed normal and $C_ie^{A_it}\xi_i$ were within 10^{-6} of zero for each value of ϵ in every example.

4.5 Two-Dimensional State Examples

We continue our analysis by examining seven two-dimensional examples. Six of these examples exhibit properties that help to shed light on the shape and energy of the minimum energy detection signal, $\widehat{v}$, as well as the shapes of the normal to the separating hyperplane, $a_\epsilon(t)$, the midpoint of the shortest line segment between the output sets, $\overline{y}_\epsilon(t)$, and the separability index, γ^*, for two-dimensional problems. The first of these six was run on seven different interval lengths in order to examine the oscillatory properties of $\widehat{v}$. The seventh example is constructed to be unobservable in order to test the algorithm's capability to handle such problems.

4.5.1 Primary Two-Dimensional Example

In Example 4.5 we find a problem for which the shape of $\widehat{v}$ varies considerably with the interval length.

EXAMPLE 4.5. *[Change of eigenvalues, neutral stability] This problem has purely imaginary eigenvalues. The only difference between models is a change in eigenvalues from $\pm 3i$ to $\pm 2i$.*

$$
\begin{aligned}
x_0' &= z_0 + \mu_3 && \text{(4.18a)}\\
z_0' &= -9x_0 + v + \mu_2 && \text{(4.18b)}\\
y &= x_0 + \mu_1 && \text{(4.18c)}\\
x_1' &= z_1 + \mu_6 && \text{(4.18d)}\\
z_1' &= -4x_1 + v + \mu_5 && \text{(4.18e)}\\
y &= x_1 + \mu_4. && \text{(4.18f)}
\end{aligned}
$$

Figure 4.11 shows $\widehat{v}$ for $t_f = 1, 20$. Figure 4.12 shows $\widehat{v}$ for $t_f = 1, 2, 4, 6, 8, 10, 20$. In the $t_f = 1$ case, $\widehat{v}$ resembles the detection signal in the one-dimensional cases studied, but at longer intervals we get a very different $\widehat{v}$. Oscillations are introduced, and the number and period of the oscillations depend on the interval length. It should be noted that the plots in Figure 4.12 are actually $\gamma^* \widehat{v}$, i.e., they were computed in formulation 1nR but not rescaled by the reciprocal of the separability index as required in that formulation (see Chapter 2). Not performing the rescaling allows for a meaningful comparison of the various shapes of $\widehat{v}$, but does not allow for a comparison of $||\widehat{v}||$ between intervals. If the plots were rescaled by $\frac{1}{\gamma^*}$, the $t_f = 1, 2$ cases would dwarf all others in magnitude, and the curve shape comparison would be impossible.

Table 4.4 shows $||\widehat{v}||$ and other quantities of interest for Example 4.5 on each

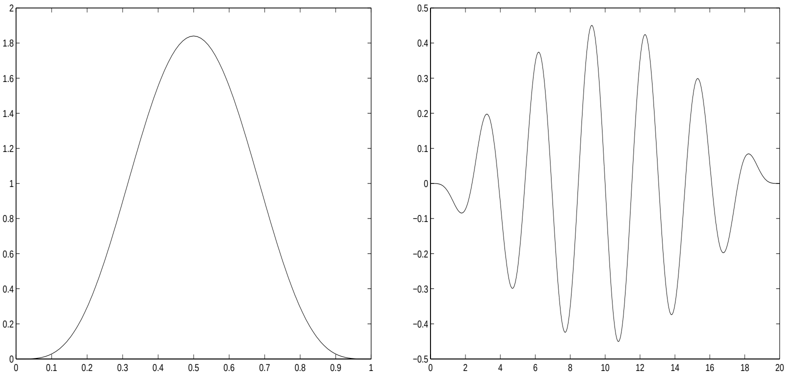

Figure 4.11: $\widehat{v}$ for Example 4.5: $t_f = 1$ (left) and $t_f = 20$ (right)

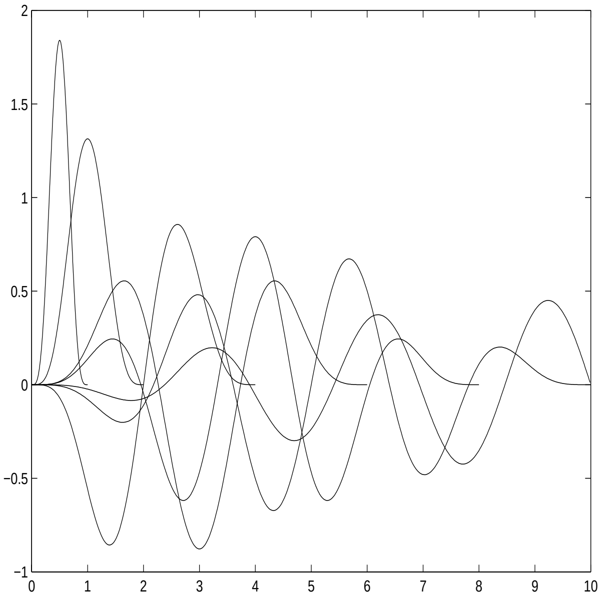

Figure 4.12: $\widehat{v}$ for Example 4.5: $t_f = 1, 2, 4, 6, 8, 10, 20$

interval tested. We used an initial mesh size of 50 for all but the $t_f = 10$ case. That case required an initial mesh of 20 in order to attain feasibility. This resulted in more iterations before convergence, but still gave CPU time comparable to the first four cases. As always, CPU times should only be used as rough indicators since they often vary greatly. As an example, note that the CPU time for the $t_f = 8$ case is an unexplainable outlier. For the $t_f = 12, 14, 16, 18$ cases we used default tolerances which are looser than the tolerances used in all other cases. This was done to demonstrate the efficiency of the software under default conditions. Note the CPU times for these cases are considerably lower than the other cases and the number of iterations have not increased. Aside from the difference in tolerances and the natural variation often present, the presence of inconsistent CPU times may indicate the need to optimize either the conditioning on the objective function or the initial mesh size for each interval.

Also note the steady decrease of β as the interval lengthens. This effect is probably due to the difference in dynamics between models becoming more significant as the detection interval lengthens.

Finally, notice that γ^* is extremely small on the shorter intervals. As the interval lengthens, however, γ^* grows to be larger than that for the one-dimensional Example 4.1. Thus, Example 4.5 requires a higher energy $\widehat{v}$ than Example 4.1 on the shorter intervals, but a lower energy $\widehat{v}$ on longer intervals. Figure 4.13 depicts the relationship of γ^* to the interval length for this example, and a comparison to γ^* from Example 4.1.

Figure 4.14 gives $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.5, for various values of ϵ, and for a typical interval length, $t_f = 6$. Contrary to what we saw in previous plots of the parameters of the separating hyperplane, these plots depict some sensitivity to the value of ϵ. This sensitivity indicates that one should not blindly choose a value of ϵ for a given problem without first determining whether the value of ϵ is

Table 4.4: Performance comparison of Example 4.5 on various time intervals

t_f	γ^*	β	CPU Time	Iterations
1	$0.72694 * 10^{-3}$	0.500104	850.94	3
2	$0.14710 * 10^{-1}$	0.501659	388.05	3
4	$0.91764 * 10^{-1}$	0.502983	867.55	5
6	0.18093	0.463663	960.15	4
8	0.23532	0.469403	4580.01	5
10	0.27321	0.404642	836.74	9
12	0.29050	0.351149	264.08	5
14	0.30296	0.315157	267.95	5
16	0.31787	0.293182	259.66	5
18	0.32954	0.262228	330.09	5
20	0.33806	0.240156	9464.59	6

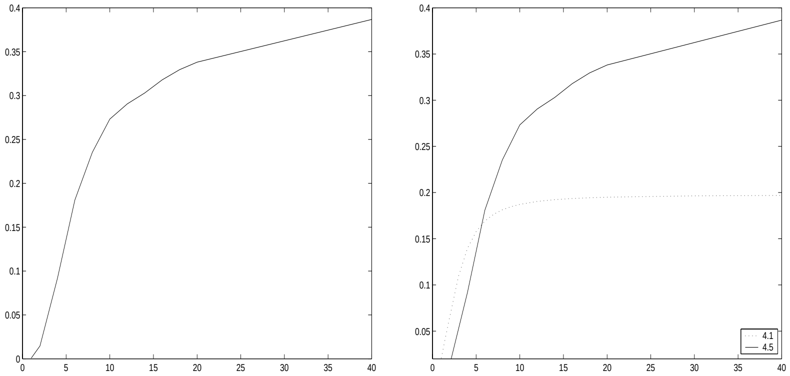

Figure 4.13: γ^* for Example 4.5 as a function of t_f (left), compared with Example 4.1 (right)

a factor. For this problem, and most likely many others, the value of ϵ affects the accuracy of the approximation to the true separating hyperplane, and thus should be chosen to minimize the error while ensuring a positive separation between output sets. Fortunately, tests thus far indicate that the calculations are reasonably robust and parameters are usually not highly sensitive to the value of ϵ. In addition, $a_\epsilon(t)$ varies smoothly with ϵ, and thus one can guarantee high quality results by experimenting with ϵ values. Ultimately, if perfect model identification is required, one should apply $\delta\widehat{v}$ with $\delta > 1$ and use $\epsilon = 1$ in the MI algorithm. Additional energy in v is a small price to pay to guarantee that $a(t)$ is accurate to within machine precision.

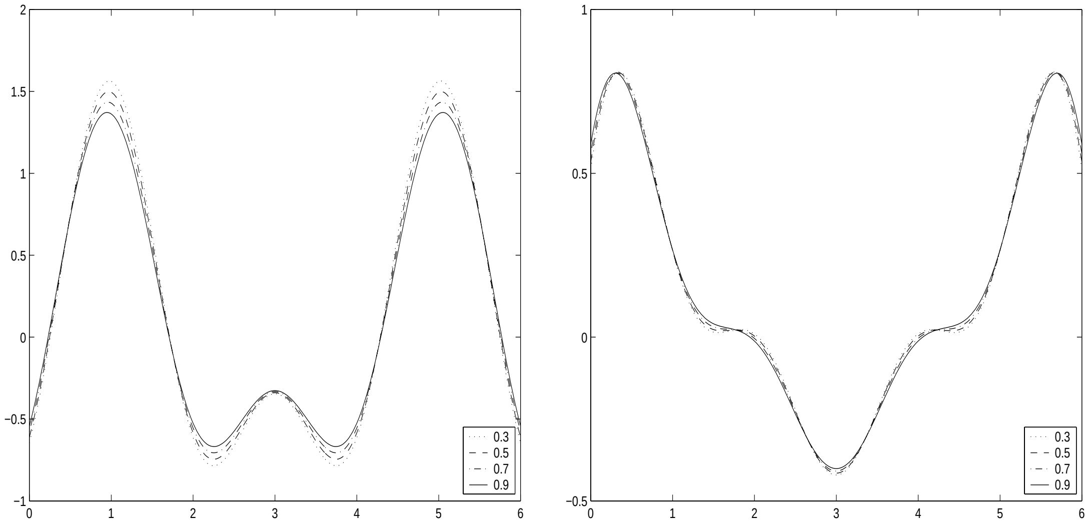

Figure 4.14: $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.5: $\epsilon = 0.3, 0.5, 0.7, 0.9$

4.5.2 Other Two-Dimensional Examples

The remaining two-dimensional examples, while exhibiting various dynamical and weighting properties, all possess similar $\widehat{v}$ and $a_\epsilon(t)$ qualities. Thus, they will be treated together.

EXAMPLE 4.6. *[Change of eigenvalues, neutral stability] This problem has the same change in eigenvalues from $\pm 3i$ to $\pm 2i$ as Example 4.5. It also has different weighting on the noise matrix, M_0.*

$$\begin{aligned} x_0' &= z_0 + 5\mu_3 && (4.19a) \\ z_0' &= -9x_0 + v + 4\mu_2 && (4.19b) \\ y &= x_0 + \mu_1 && (4.19c) \\ x_1' &= z_1 + \mu_6 && (4.19d) \\ z_1' &= -4x_1 + v + \mu_5 && (4.19e) \\ y &= x_1 + \mu_4. && (4.19f) \end{aligned}$$

EXAMPLE 4.7. *[Change of eigenvalues, neutral-to-unstable] This problem has eigenvalues which change from purely imaginary, $\pm 3i$, to complex unstable, $\frac{1}{10} \pm 3i$.*

$$\begin{aligned} x_0' &= z_0 + \mu_3 && (4.20a) \\ z_0' &= -9x_0 + v + \mu_2 && (4.20b) \\ y &= x_0 + \mu_1 && (4.20c) \\ x_1' &= z_1 + \mu_6 && (4.20d) \\ z_1' &= -9.01x_1 + \frac{1}{5}z_1 + v + \mu_5 && (4.20e) \\ y &= x_1 + \mu_4. && (4.20f) \end{aligned}$$

Example 4.8. *[Change of eigenvalues, neutral-to-stable] This problem has eigenvalues which change from purely imaginary, $\pm 3i$, to complex stable, $-\frac{1}{10} \pm 3i$.*

$$x_0' = z_0 + \mu_3 \tag{4.21a}$$

$$z_0' = -9x_0 + v + \mu_2 \tag{4.21b}$$

$$y = x_0 + \mu_1 \tag{4.21c}$$

$$x_1' = z_1 + \mu_6 \tag{4.21d}$$

$$z_1' = -9.01x_1 - \frac{1}{5}z_1 + v + \mu_5 \tag{4.21e}$$

$$y = x_1 + \mu_4. \tag{4.21f}$$

Example 4.9. *[No change of eigenvalues, neutral stability] This problem has no change in dynamics. Both models have eigenvalues of $\pm 3i$. The difference between models is a change in weighting of the noise matrix, M_1, and the output matrix, C_1.*

$$x_0' = z_0 + \mu_3 \tag{4.22a}$$

$$z_0' = -9x_0 + v + \mu_2 \tag{4.22b}$$

$$y = x_0 + \mu_1 \tag{4.22c}$$

$$x_1' = z_1 + 2\mu_6 \tag{4.22d}$$

$$z_1' = -9x_1 + v + 3\mu_5 \tag{4.22e}$$

$$y = 5x_1 + \mu_4. \tag{4.22f}$$

Example 4.10. *[Change of eigenvalues, stable-to-stable] This problem has eigenvalues which change from complex stable, $-1 \pm 3i$, to complex stable, $-2 \pm 2i$.*

$$\begin{aligned} x_0' &= -x_0 + z_0 + \mu_3 && \text{(4.23a)} \\ z_0' &= -9x_0 - z_0 + v + \mu_2 && \text{(4.23b)} \\ y &= x_0 + \mu_1 && \text{(4.23c)} \\ x_1' &= -2x_1 + z_1 + \mu_6 && \text{(4.23d)} \\ z_1' &= -4x_1 - 2z_1 + v + \mu_5 && \text{(4.23e)} \\ y &= x_1 + \mu_4. && \text{(4.23f)} \end{aligned}$$

On the left side of Figure 4.15 is $\widehat{v}$ on the interval $[0, 1]$ for Examples 4.5-4.10. Some scaling problems exist among the examples on this interval, so this plot is left unrescaled as before. Note that, for the first time thus far, we have some oscillations in $\widehat{v}$ on our shortest interval. This indicates that, for those problems which exhibit the oscillation, $[0, 1]$ is a long interval relative to their dynamics. This result is not surprising, as we would expect the dynamics of the problem to influence the affect of a given test period length on the detection signal.

The right side of Figure 4.15 shows the same information on the interval $[0, 10]$. Here, the scaling problem is not as severe, so each $\widehat{v}$ has been rescaled by $\frac{1}{\gamma^*}$. Note that one problem, Example 4.9, exhibits a single-lobed $\widehat{v}$, in contrast to all other two-dimensional problems so far on this interval. Recall that this example has no change in dynamics between models, but only various different weight matrices. Thus we should expect for this type of problem that $\widehat{v}$ merely has to contain enough energy to drive the output sets apart without requiring any complex wave form, as in the one-dimensional problems examined previously. Figure 4.16 gives all examples exhibiting a simple $\widehat{v}$ for the case $t_f = 10$. These plots are all scaled properly, and the highest energy plot is that of Example 4.9.

As we mentioned in the previous section, [29] observed that the $\widehat{v}$ seem to lie in a region bounded by half of a period of the sine function. Figure 4.17 shows this by plotting the same information as Figure 4.15 (right), with each curve normalized in energy. It is quite apparent that all oscillations of the various $\widehat{v}$ do occur in an envelope that resembles the sine function.

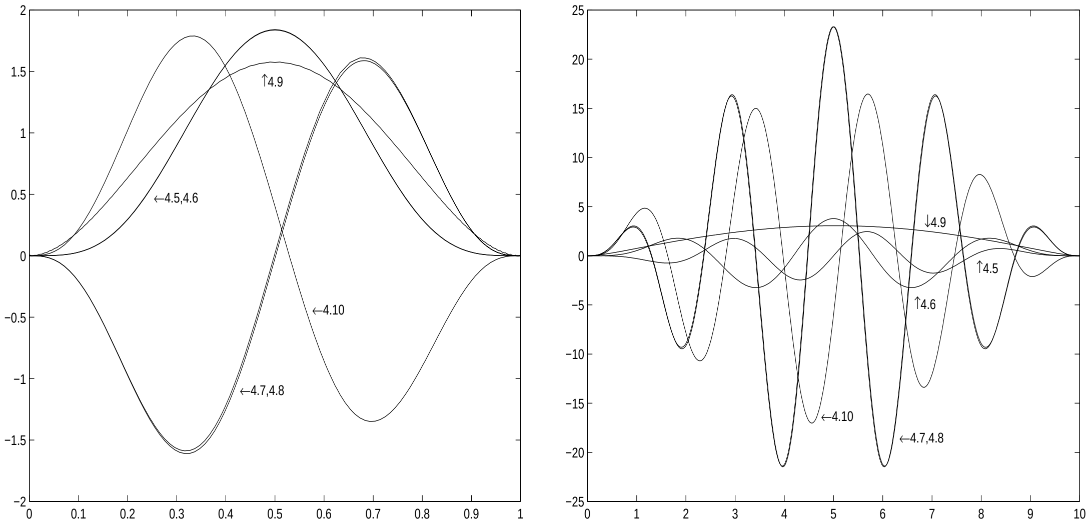

Figure 4.15: $\widehat{v}$ for Examples 4.5-4.10: $t_f = 1$ (left), $t_f = 10$ (right)

Table 4.5 compares various quantities from Examples 4.2-4.4 and 4.5-4.10 for $t_f = 1$, while Table 4.6 does the same for $t_f = 10$. Again, an initial mesh of 50 was used for all examples, except Examples 4.2, 4.5, 4.9, and 4.10 on $[0, 10]$ used an initial grid of 20 for the feasibility reasons mentioned above. It is clear from Table 4.5 that the one-dimensional examples, along with the two-dimensional example with unchanging dynamics, possess higher separability indices than the two-dimensional examples with changing dynamics, and thus admit a lower energy $\widehat{v}$. This difference

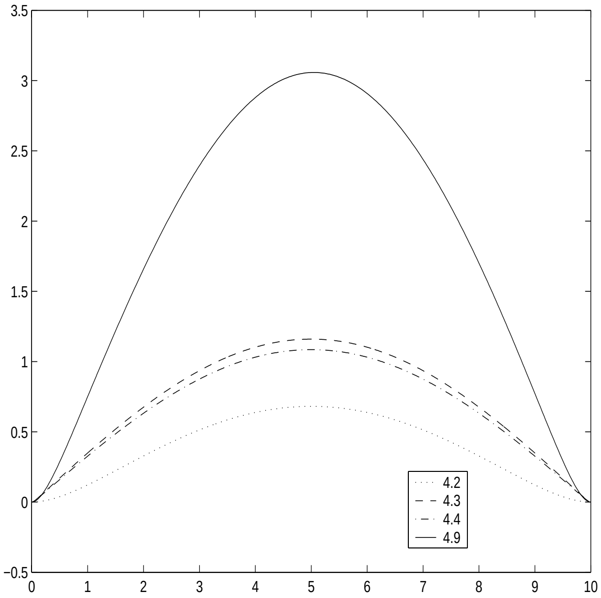

Figure 4.16: $\widehat{v}$ for Examples 4.2,4.3,4.4,4.9: $t_f = 10$

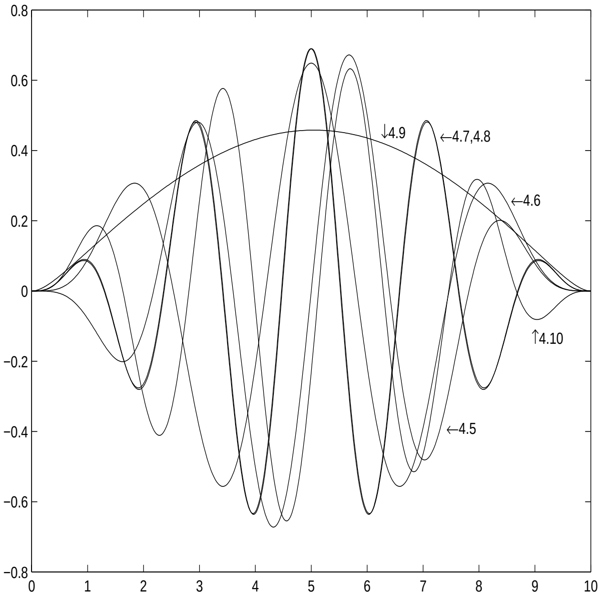

Figure 4.17: Normalized $\widehat{v}$ for Examples 4.5-4.10: $t_f = 10$

is somewhat clouded as the interval lengthens to 10, indicating that, on longer intervals, the energy required in $\widehat{v}$ becomes more dependent on system dynamics than on problem dimension or simplicity.

Table 4.5: Performance comparison of Examples 4.2-4.10: $t_f = 1$

Example	γ^*	β	CPU Time	Iterations
4.2	$0.32606 * 10^{-1}$	0.499823	38.69	2
4.3	0.13418	0.509624	310.90	4
4.4	0.14344	0.509624	234.53	4
4.5	$0.72694 * 10^{-3}$	0.500104	850.94	3
4.6	$0.69719 * 10^{-3}$	0.520423	829.42	3
4.7	$0.13485 * 10^{-3}$	0.499998	1535.48	3
4.8	$0.13485 * 10^{-3}$	0.499998	533.84	3
4.9	$0.64642 * 10^{-1}$	0.314308	795.85	3
4.10	$0.12853 * 10^{-2}$	0.500149	870.98	3

Table 4.6: Performance comparison of Examples 4.2-4.10: $t_f = 10$

Example	γ^*	β	CPU Time	Iterations
4.2	0.69849	0.418685	31.37	4
4.3	0.38687	0.579995	838.09	7
4.4	0.41355	0.579995	1281.74	8
4.5	0.27321	0.404642	836.74	9
4.6	0.17170	0.529625	8490.14	5
4.7	$0.29614 * 10^{-1}$	0.504179	7165.95	8
4.8	$0.29614 * 10^{-1}$	0.504179	2657.45	5
4.9	0.14981	0.335714	2971.26	6
4.10	$0.38458 * 10^{-1}$	0.514021	469.30	7

Figure 4.18 gives a typical $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for these examples.

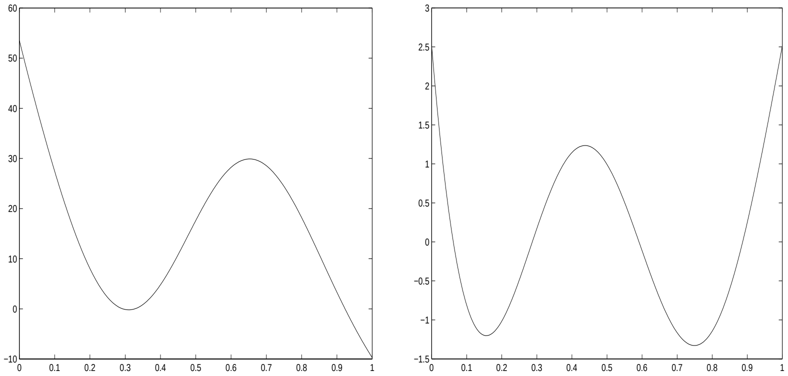

Figure 4.18: $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.10: $\epsilon = 0.7$

4.5.3 Common Mode Two-Dimensional Example

For our last two-dimensional example we present a problem in which the two models share a common eigenvalue, eigenvector pair.

Example 4.11. *[Shared eigenvalue, unstable-to-unstable] This problem has a change in its first eigenvalue from* 2 *to* 3. *The other eigenvalue does not change between models. It remains at* 1.

$$x_0' = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} x_0 + \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} v + \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mu_0 \tag{4.24a}$$

$$y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_0 + \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \mu_0 \tag{4.24b}$$

$$x_1' = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} x_1 + \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} v + \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mu_1 \tag{4.24c}$$

$$y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_1 + \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \mu_1 . \tag{4.24d}$$

Both models of this example are controllable and observable. To see this fact, recall from Chapter 1 that a second-order system is controllable if and only if $[sI - A \mid B]$ has rank 2 for all values of s. The same system is observable if and only if $[sI - A^T \mid C^T]$ has rank 2 for all values of s [6]. Clearly, both models of Example 4.11 are controllable since B_0 and B_1 are full rank by themselves. Both models are also observable since the same holds for C_0^T and C_1^T. However, the fact that there exist x_0 and x_1 subspaces on which the dynamics and the outputs are the same is equivalent to saying that the combined system is not observable. These subspaces exist because

$$x = \begin{pmatrix} e^t \xi_{i0} \\ 0 \end{pmatrix} \tag{4.25}$$

is a solution of the free response for both of them. Thus each output set has a parallel side in the long direction.

In terms of the FDMI algorithm, as it attempts to separate the output sets in the detection signal phase, the last points to touch will be those on the parallel side. Many states and outputs will be eligible to be part of the optimal solution, and because of this nonuniqueness, the algorithm may be unable to choose one. Simply stated, the two models may not be *different* enough for the algorithm to find a unique minimal v. Even if it is possible to choose an optimal solution from the equivalent candidates in the detection signal phase, the model identification phase also separates the output sets. As the algorithm searches for the closest points on the closure of each set, it will again find many eligible points on each parallel side, and may be unable to choose.

In fact, this problem does occur in Example 4.11. As mesh refinements are made and the dynamics are resolved to near-tolerance accuracy, the algorithm terminates with a warning about degenerate constraints and SQP errors. (When the problem is modified to remove the parallel sides, all errors disappear and the problem converges to any desired tolerance.) Despite the errors, the dynamics are resolved quite accurately, down to the 10^{-6} level. In addition, the algorithm is converging to a solution before the errors occur. So it appears that the algorithm is able to handle this type of problem despite the lack of uniqueness of the optimal solution. Figures 4.19 and 4.20 give $\widehat{v}$, $\overline{y}_\epsilon(t)$, and $a_\epsilon(t)$ for this problem on the interval $[0, 5]$.

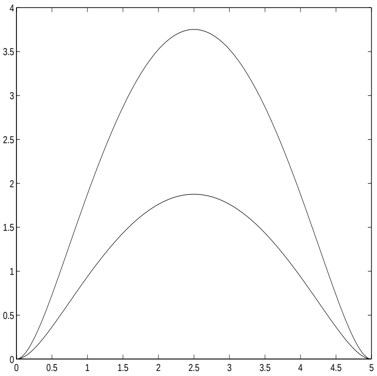

Figure 4.19: Components of $\widehat{v}$ for Example 4.11: $t_f = 5$

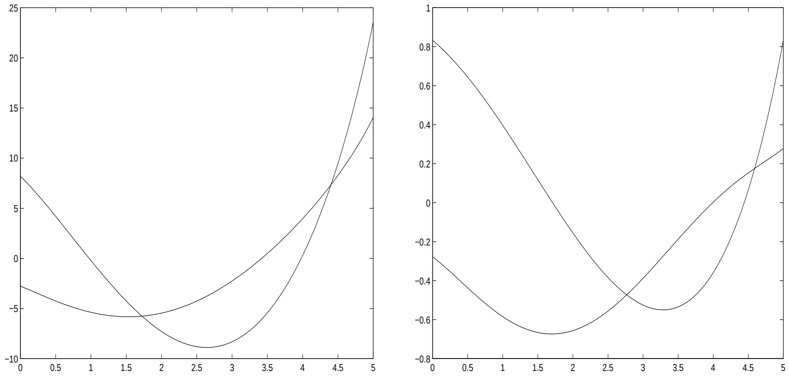

Figure 4.20: Components of $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.11: $\epsilon = 0.7$

If the FDMI algorithm is not able to solve a given problem with common eigenvalue, eigenvector pairs between models, we have the option of truncating the outputs. That is, we multiply outputs from both models by the same matrix, annihilating the identical part, and solve the resulting lower dimensional problem as before. Testing of this procedure will be left to future research.

4.6 Industrial Example

We now turn to a real world example in order to test the performance of the FDMI algorithm on larger scale problems. The problem is the equalized and linearized model of a single-engine F-16 aircraft [39]. The model has a three-dimensional state, and the dynamics include a reference control input.

EXAMPLE 4.12. *[Change of eigenvalues, stable-to-unstable] The nominal model of this problem has one stable real eigenvalue, and two stable complex eigenvalues. The fault model has three unstable real eigenvalues.*

$$x_0' = \begin{bmatrix} -0.1689 & 0.0759 & -0.9952 \\ -26.859 & -2.5472 & 0.0689 \\ 9.3603 & -0.1773 & -2.4792 \end{bmatrix} x_0 + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} w \\ u \end{pmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mu_0 \tag{4.26a}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.9971 & 0.0755 \end{bmatrix} x_0 + \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \mu_0. \tag{4.26b}$$

A fault model simulating an electrical interruption to a flight control computer's input channels may be represented as

$$x_1' = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} x_1 + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} w \\ u \end{pmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mu_1 \tag{4.26c}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.9971 & 0.0755 \end{bmatrix} x_1 + \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \mu_1. \tag{4.26d}$$

The three states are side-slip, roll rate, and yaw rate, and the control vector is

$$\begin{pmatrix} w \\ u \end{pmatrix} = \begin{pmatrix} \textit{rudder input} \\ \textit{stick input} \end{pmatrix}.$$

Assuming that the detection signal is applied on the same channels as the control vector during the test period, one of the following occurs:

- the control is nulled so that only the detection signal is inputted,
- the control remains. It is subtracted from $\widehat{v}$ from the MEDS algorithm to obtain the additional signal required to separate the output sets of the two models.

The second option is the equivalent of solving one of the alternative problems described in Chapter 2, i.e.,

$$\min \|v\| \text{ subject to } \max_{0<\beta<1} J_{u+v}(\beta) \geq 1. \tag{4.27}$$

If the detection signal must be kept off of the control channels, then the other problem described in Chapter 2 is solved, i.e.,

$$\min \|w_1\| \text{ subject to } \max_{0<\beta<1} J_w(\beta) \geq 1 \text{ and } w_2 = u. \tag{4.28}$$

where the dynamics are described by

$$x_i' = A_i x_i + B_i v + E_i u + M_i \mu_i. \tag{4.29}$$

and where $w = [w_1, w_2] = [v, u]$. Obviously, if the nulling option is chosen, the test period should be kept as short as possible to avoid aircraft control difficulties. The results below reflect the nulling option.

SOCS has no difficulty solving this problem. The FORTRAN code generated is lengthy, but optimized coding methods can minimize the impact of this effect. In fact, standard routines for translating matrix multiplication into FORTRAN loops can reduce the effect of the added dimensionality to virtually zero. Figures 4.21 and 4.22 give $\widehat{v}$, $\overline{y}_\epsilon(t)$, and $a_\epsilon(t)$ for this problem on the interval $[0, 1]$.

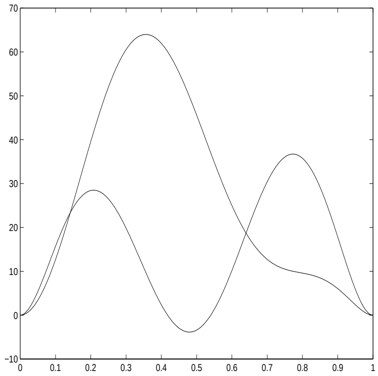

Figure 4.21: Components of $\widehat{v}$ for Example 4.12: $t_f = 1$

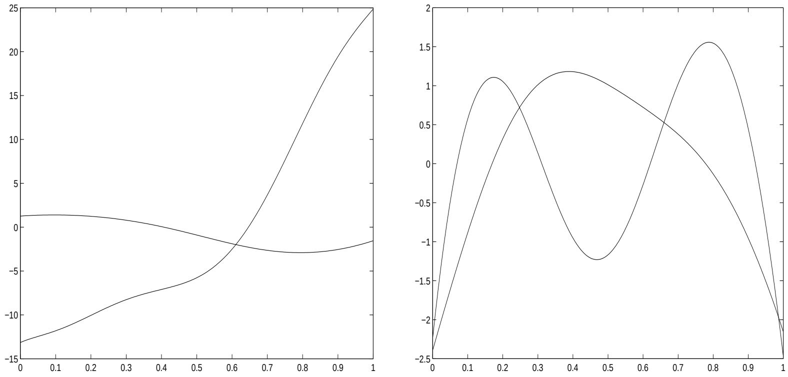

Figure 4.22: Components of $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.12: $\epsilon = 0.7$

These results demonstrate that the FDMI algorithm is capable of handling higher dimensional problems and is limited only by the multi-model assumptions inherent in the approach.

4.7 Multiple Fault Model Examples

To conclude the analysis of examples, we examine two systems, each of which has a nominal model and two fault models. One of these systems is one-dimensional and the other is two-dimensional.

Recall from Chapter 2 the two approaches described to handle multiple fault model problems. The first method is sequential in nature, in which the test period is divided into equal segments, and problems involving each pair of models are solved on the shorter intervals. It was suggested that this method would produce a $\widehat{v}$ that was larger than necessary. The second method should produce a $\widehat{v}$ of much lower energy. It is simultaneous in nature, in which the pairwise problems are solved independently on the full interval for a common $\widehat{v}$.

Our last two examples demonstrate the ability of the FDMI algorithm to handle multiple fault model systems as well as shed light on the conjecture of Chapter 2 about the energy of the $\widehat{v}$ resulting from each method.

4.7.1 One-Dimensional Example

The simplest multiple fault model problem is given in Example 4.13.

EXAMPLE 4.13. *[Change of eigenvalue, stable/stable/stable] This problem has a stable nominal model and two stable fault models. Eigenvalues are* -1*,* -3*, and* -0.2*,* *respectively.*

$$\begin{aligned} \textit{model 0: } x_0' &= -x_0 + v + \mu_2 && \text{(4.30a)} \\ y &= x_0 + \mu_1 && \text{(4.30b)} \\ \textit{model 1: } x_1' &= -3x_1 + v + \mu_4 && \text{(4.30c)} \\ y &= x_1 + \mu_3 && \text{(4.30d)} \\ \textit{model 2: } x_2' &= -0.2x_2 + v + \mu_6 && \text{(4.30e)} \\ y &= x_2 + \mu_5. && \text{(4.30f)} \end{aligned}$$

Figure 4.23 gives $\widehat{v}$ for the two methods. On the left is the comparison of $\widehat{v}$ using the sequential method on $[0, 1]$, $[1, 2]$, and $[2, 3]$ to that of the simultaneous method on $[0, 3]$. The plots are in the same scale, and it is obvious that the simultaneous method produces a $\widehat{v}$ of much lower energy. On the right side of Figure 4.23 is a comparison of $\widehat{v}$ from the simultaneous method to each $\widehat{v}$ obtained by solving the pairwise problems completely independently on the entire interval $[0, 3]$. For this problem, the simultaneous $\widehat{v}$ is the same as the highest energy $\widehat{v}$ from the independent pairwise problems. Thus, we can handle multiple fault models for the same energy and with the same $\widehat{v}$ wave form as required for one fault model. Often, however, the simultaneous $\widehat{v}$ will require more energy and will not be the same shape as the any of the three independent $\widehat{v}$. However, the similar shapes of the $\widehat{v}$ for this problem suggest that if feasibility issues arise in the simultaneous method, the $\widehat{v}$ from the pairwise problems, or multiples thereof, can be applied as an accurate initial guess for the combined problem.

Figure 4.24 gives the $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for the sequential method, and Figure 4.25 gives the same for the simultaneous method. The only immediate advantages which can be gleaned for the simultaneous method from these plots is the continuity of $\overline{y}$ and

a over the entire interval, and the lack of a requirement to maintain three different test functions. Since $\overline{y}$ and a must be stored, if a number of comparisons between multi-dimensional models must be made, the reduced number of test functions demonstrates better use of possibly limited memory resources.

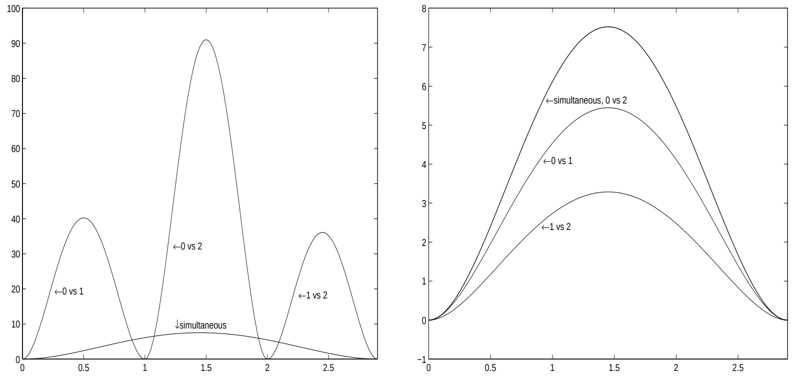

Figure 4.23: $\widehat{v}$ for Example 4.13 sequential vs. simultaneous (left), full interval two-model vs. simultaneous (right)

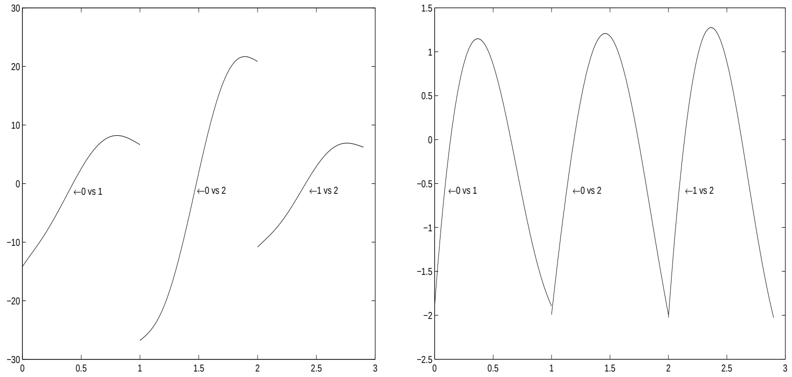

Figure 4.24: $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.13: sequential solve

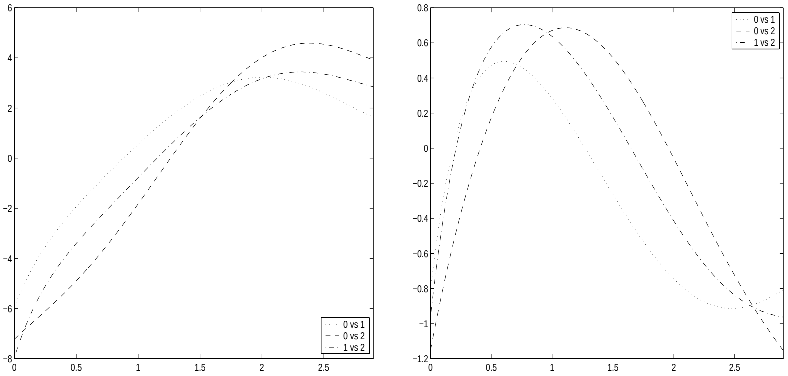

Figure 4.25: $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.13: simultaneous solve

4.7.2 Two-Dimensional Example

Our final example exhibits several interesting stability properties.

Example 4.14. *[Change of eigenvalues, stable/stable/unstable] This problem has a stable nominal model, one stable fault model, and one unstable fault model. Nominal model eigenvalues are* $-2 \pm i$. *Fault model* 1 *eigenvalues are* -9.7016 *and* -3.2984. *Fault model* 2 *eigenvalues are* $\frac{1}{2} \pm 1.3229i$.

$$\textit{model 0: } x_0' = \begin{bmatrix} -1 & 2 \\ -1 & -3 \end{bmatrix} x_0 + \begin{bmatrix} 1 \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \mu_0 \tag{4.31a}$$

$$y = \begin{bmatrix} 1 & 1 \end{bmatrix} x_0 + \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mu_0 \tag{4.31b}$$

$$\textit{model 1: } x_1' = \begin{bmatrix} -10 & 2 \\ -1 & -3 \end{bmatrix} x_1 + \begin{bmatrix} 1 \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \mu_1 \tag{4.31c}$$

$$y = \begin{bmatrix} 1 & 1 \end{bmatrix} x_1 + \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mu_1 \tag{4.31d}$$

$$\textit{model 2: } x_2' = \begin{bmatrix} 1 & 2 \\ -1 & 0 \end{bmatrix} x_2 + \begin{bmatrix} 1 \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \mu_2 \tag{4.31e}$$

$$y = \begin{bmatrix} 1 & 1 \end{bmatrix} x_2 + \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mu_2. \tag{4.31f}$$

Figure 4.26 gives $\widehat{v}$ for the two methods. On the left is the comparison of $\widehat{v}$ using the sequential method to that of the simultaneous method. The plot oscillating minimally around the horizontal axis is that of the simultaneous method. Again it is obvious that the simultaneous method produces a $\widehat{v}$ of much lower energy. On the right side of Figure 4.23 is a comparison of $\widehat{v}$ from the simultaneous method (dotted line) to the $\widehat{v}$ obtained by solving the pairwise problems completely independently on the entire interval (solid lines). The simultaneous $\widehat{v}$ is not the same as any other $\widehat{v}$ from the independent pairwise problems. In fact, the simultaneous $\widehat{v}$ contains an extra extrema on the interval. Even so, the plot shows that we can handle multiple fault models with only a bit more energy than required for a single fault model.

Figure 4.27 gives the $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for the sequential method, and Figure 4.28 gives the same for the simultaneous method. Again, the only advantages appear to be the continuity of $\overline{y}$ and a, and simplicity of the test function for the simultaneous method.

4.8 Conclusion

The purpose of this chapter was to give examples and analysis of the practical aspects of the FDMI algorithm. After restating the problem and one form of the algorithm, we examined the simplest problems in one dimension to ascertain shapes and energies of the minimum energy detection signals for those problems. Comparisons of one problem in four different formulations using different initial guesses showed that each arrived at the same optimal solution, supporting uniqueness assertions made in Chapter 3. We then looked at more complex examples in two dimensions to demonstrate how the basic shapes and energies of $\widehat{v}$ evolve. An unobservable system was tested, and despite a known lack of uniqueness of the closest points in the output sets, the

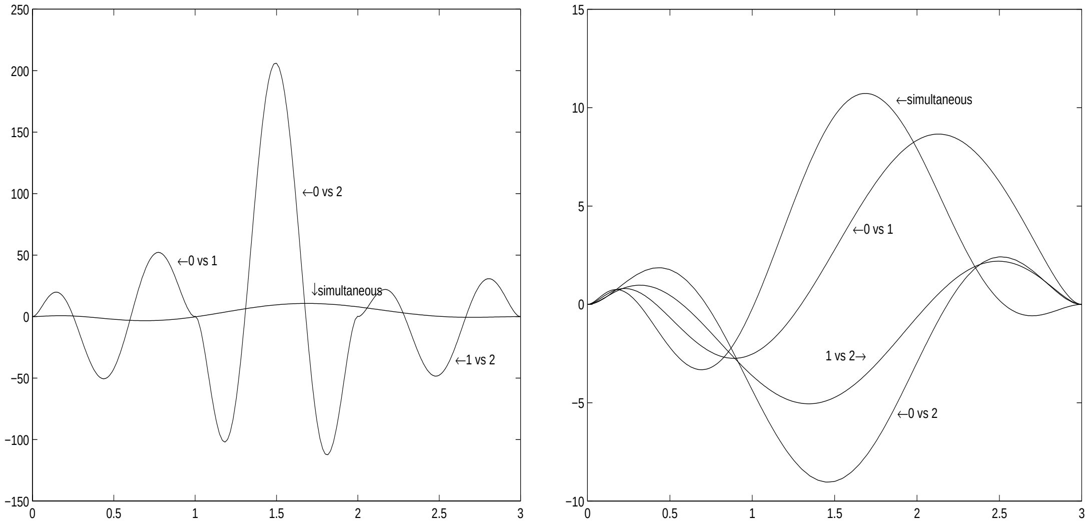

Figure 4.26: $\widehat{v}$ for Example 4.14 sequential vs. simultaneous (left), full interval two-model vs. simultaneous (right)

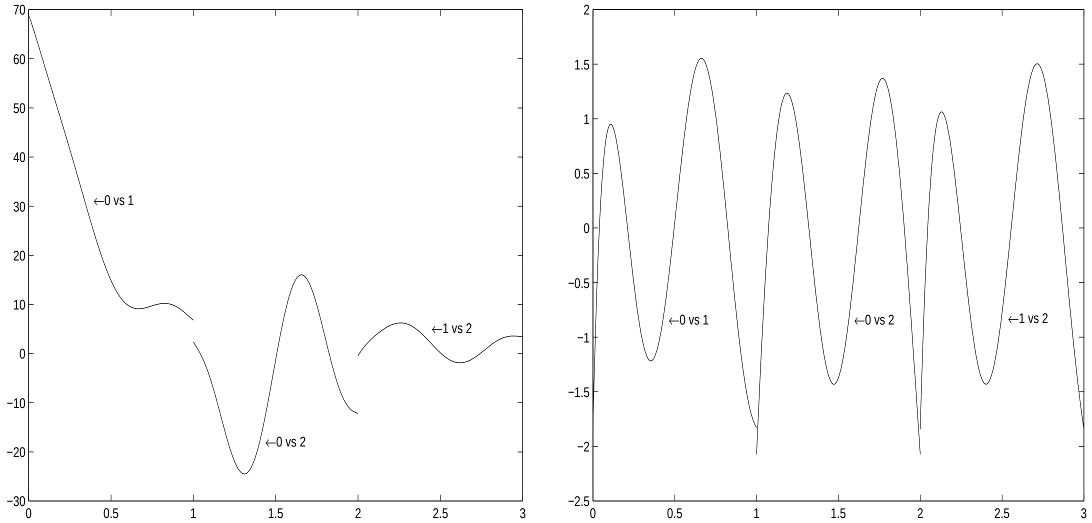

Figure 4.27: $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.14: sequential

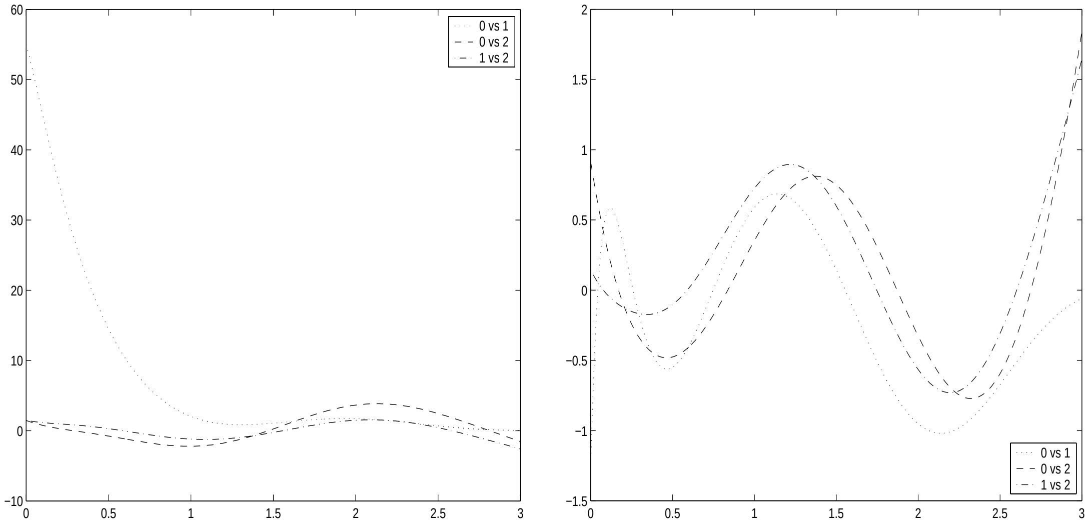

Figure 4.28: $\overline{y}_\epsilon(t)$ and $a_\epsilon(t)$ for Example 4.14: simultaneous

FDMI algorithm and SOCS provided a solution. An industrial strength example in three dimensions proved that the algorithm and SOCS were capable and efficient even in the higher dimensions. We ended with examples of the extension of the algorithm to problems with multiple fault models.

Throughout the examples we saw varying degrees of sensitivity to the value of ϵ. It was demonstrated that some care must be taken in choosing the value of ϵ used in the algorithm. However, it was also shown that high quality results can be reliably computed with the proper selection of ϵ. In addition, we saw some difficulties arising due to very small objective function values. A method of automatic conditioning on objective functions should be developed to overcome these difficulties. Finally, systems for which a feasible initial guess is difficult to obtain due to initial mesh size considerations may require pre-conditioning to facilitate making the initial guess. This problem may also arise due to the fact that $-\widehat{v}$ and $\widehat{v}$ are both optimal solutions. An initial guess of $v = 0$ may be suitable for some problems, but it is a saddle point, and using it could slow down convergence of the SOCS program. Despite these issues, the analysis in this chapter demonstrates the efficiency and practicality of the FDMI algorithm.

Chapter 5

Future Work and Conclusions

5.1 Future Work

Up to this point, we have exercised the FDMI algorithm mostly within the confines of certain assumptions about the structure of the problem to be solved. For example, we have assumed a short test period length, no *a priori* knowledge of initial conditions on the state, and strict linearity in the dynamics. While we have extended the algorithm to problems with multiple fault models, with a pre-existing known control, and with alternative cost functions, we have not mentioned other variations that may help to extend the algorithm to an even larger set of problems. Some variations we have mentioned, but have left to future research. The unreduced model, mentioned in Chapter 2, would be of direct benefit to several types of problems excluded from our study. Also, unobservable systems, theoretically excluded from the FDMI algorithm, may in fact be treatable. While we discussed the possibility of applying the FDMI algorithm to such systems, and even worked an example for which the algorithm provided a solution, the bulk of the work remains. In particular, the most promising aspect, that of projecting each model onto a subspace to eliminate the parallel sides, is left undone.

This chapter will introduce and discuss several other interesting and applicable variations to the basic problem and the impact they have on the FDMI algorithm. Topics mentioned below are: the half-infinite interval, linear time varying problems, nonlinear problems, independent noise channels, and sensitivity issues. We mention these topics for the purpose of highlighting the work still remaining in this area, and thus will not attempt a complete and detailed look at each one. Conclusions follow the last topic and complete our discussion.

5.1.1 The Half-Infinite Interval

We mentioned in Chapter 1 that Nikoukhah *et al.* [29] was the inspiration for the work contained in this thesis. In that paper, theory is developed for the limiting shape and energy of $\widehat{v}$ on the interval $[0, \infty)$. While in practice the use of the half-infinite detection horizon is not always possible due to unstable fault models or cost considerations, theoretical development of the limiting case can aid in approximating detection signals and separability indices on long intervals. As this thesis is dedicated more to practical applications of online detection, we leave the development of the limiting case to future research.

To aid in that research, we note that Riccati matrix differential equations provide convenient properties on, and useful insights into the half-infinite interval. Finite interval Riccati differential equations become algebraic equations as t_f goes to infinity. In fact, a large part of the Riccati form of the FDMI algorithm should be directly extendable to the half-infinite interval. The translation of other parts and confirmation of the results remain to be accomplished. In particular, questions remain about the multiple fault model case, and whether the FDMI algorithm will be capable of addressing those problems on the half-infinite interval.

5.1.2 Linear Time Varying Models

The linear time invariant problem addressed thus far is a special case of a larger set of control problems, the linear time varying (LTV) problem. In the LTV case, linearity still exists, but the coefficient matrices A, B, C, M, and N depend on time. Thus the multi-model system model 4.1 becomes

$$\begin{aligned} x_i' &= A_i(t)x_i + B_i(t)v + M_i(t)\mu_i && \text{(5.1a)} \\ y &= C_i(t)x_i + N_i(t)\mu_i && \text{(5.1b)} \end{aligned}$$

for $i = 0, ..., m$. Luckily, convexity still exists due to linearity, so the same visualizations and characterizations of system dynamics and outputs can be made. In fact, even much of the Riccati theory mentioned in the previous section holds. Unfortunately, the infinite interval algebraic Riccati equation does not hold, so further research not based on our Riccati approach will be required for that case.

We should also note that the system reductions described in Chapter 2 will now involve time varying matrices, so software must be chosen which can handle the new complexities involved. In addition, technical difficulties arise due to time-varying coordinate changes, $x = Q(t)w$. Q' will enter the equations, and differentiation of computed quantities is highly undesirable. Also, even if N_i is full row rank there still may not exist a submatrix which is invertible for all t. These difficulties highlight the need to develop a theory and algorithms that work on systems in their original, unreduced form. It is apparent that much adaptation of theory, further development of algorithms, and tests of new examples are required before the LTV subject can be considered closed.

5.1.3 Nonlinear Models

Another larger set of control problems of which the LTI problem is a special case is the nonlinear problem. The semi-explicit nonlinear control problem can be written as

$$\begin{aligned} x' &= f[x(t), u(t), t] && (5.2a) \\ y &= g[x(t), u(t), t]. && (5.2b) \end{aligned}$$

While the FDMI algorithm should, in principle, be extendable to these types of systems, technical and computational difficulties arise due to the loss of convexity. As a result, solutions are only locally optimal, if they can be shown to exist at all, and the implementation of the algorithm in optimization codes becomes subject to significant initial guess and convergence issues.

Of the general class of nonlinear control problems, three types show promise for more immediate application of the FDMI algorithm. These are: small bounded nonlinearities, nonlinearities in only the control, and nonlinearities involving coefficient matrices dependent on v. We discuss each of them in turn.

Small Nonlinearities

Small norm-bounded nonlinearities do not present undue difficulties to the FDMI algorithm. Suppose that the models and noise bounds are of the form

$$\begin{aligned} x_i' &= A_i x_i + g_i(x_i, t) + B_i v + M_i \mu_i && (5.3a) \\ y &= C_i x_i + N_i \mu_i && (5.3b) \\ ||\mu_i||^2 &< 1. && (5.3c) \end{aligned}$$

If $||g_i(x_i,t)|| \leq \epsilon$, then we can address (5.3) by considering

$$\begin{aligned} x_i' &= A_i x_i + B_i v + \overline{M}_i \overline{\mu}_i && \text{(5.4a)} \\ y &= C_i x_i + \overline{N}_i \overline{\mu}_i && \text{(5.4b)} \\ ||\overline{\mu}_i||^2 &< 1 + \epsilon^2 t_f && \text{(5.4c)} \end{aligned}$$

where $\overline{\mu} = \begin{pmatrix} \widehat{\mu}_i \\ \mu_i \end{pmatrix}$, $\overline{M}_i = [I\ M_i]$, and $\overline{N}_i = [0\ N_i]$. After rescaling, (5.4) is in the form required by the FDMI algorithm. This formulation will produce an answer which over estimates the required $||\widehat{v}||$. Many robust algorithms can handle small nonlinearities in this way. Tests of the efficiency of the $\widehat{v}$ produced via this formulation are left to future research.

Nonlinear in the Control

Another way nonlinearities may enter the problem is through the control. Recall from Chapters 2 and 3 that [30] begins with our problem and assumptions, but goes in quite a different direction to develop the detection signal and separating hyperplane. While that approach seems more direct and satisfying, it is not capable of handling nonlinear controls. The FDMI algorithm gains a distinct advantage in this aspect because it can solve problems with nonlinearities in the control.

Suppose that the nonlinearly controlled models are

$$\begin{aligned} x_i' &= A_i x_i + B_i g(v) + M_i \mu_i, && \text{(5.5a)} \\ y &= C_i x_i + N_i \mu_i && \text{(5.5b)} \end{aligned}$$

where v may be a steering angle, for example, and thus would enter the equations as $\cos v$. The 2nR form of the optimization problem is now

$$\min_v ||v|| \text{ such that } J_{g(v)}(\beta) \geq 1 \text{ for some } \beta \in [0,1] \tag{5.6}$$

which, after the reductions described in Chapter 2, becomes a nonlinear boundary value problem. While SOCS has excellent nonlinear capabilities, the nature of $g(v)$ will have a lot to say about the solution to the BVP. If the function is not one to one, as $\cos v$ is not, then we should expect multiple local minima and saddle points. The rate of convergence to, and the quality of the solution then become a function of the initial guess. Development of an efficient initial guess algorithm becomes paramount, and is a ripe topic for future research.

Coefficient Matrices Dependent on the Detection Signal

An interesting class of nonlinear problems involves system coefficient matrices which can be affected by the detection signal. The system models for this class are

$$x_i' = A_i(v)x_i + B_i(v)v + M_i(v)\mu_i \tag{5.7a}$$

$$y = C_i(v)x_i + N_i(v)\mu_i. \tag{5.7b}$$

For this problem, for a given v the output sets, $\mathcal{A}^i(v)$, are still convex. However, they vary nonlinearly with v. Given that one has a proper detection signal, the FDMI algorithm can still be used to solve for the normal to the separating hyperplane. Obtaining the minimum energy proper detection signal may not be as straightforward. Since the minimization of $||\widehat{v}||$ occurs on the outside of all other operations, the algorithm may work as coded in SOCS. On the other hand, this type of problem is similar to the LTV problem, but with an added nonlinear structure. The difficulties described for that problem are applicable here as well. Reductions involving nonlinear v-varying matrices will undoubtedly be more complex than those involving linear time varying matrices, and will require algorithms and software that can handle their complexities. In addition, if a v-varying coordinate change is required, then v' may enter the equations. As in the LTV case, one should avoid differentiating a computed quantity. Each of these issues provides a strong argument for developing algorithms based on the unreduced problem. As in the LTV problem, adapting the algorithm, confirming the theory, and testing examples remains as future research.

5.1.4 Independent Noise Bounds

Systems may have an added complexity in that noise channels are independently bounded. Thus, the expression for the bound on the noise changes from

$$||\mu_i||^2 < 1, \quad i = 0, 1 \tag{5.8}$$

to

$$||\mu_{ij}||^2 < 1, \quad i = 0, 1, \quad j = 1, ...n_i \tag{5.9}$$

where j is the channel and n_i is the number of noise channels in model i. This additional complexity has a far reaching impact on the problem. First, (5.9) is equivalent to $||\mu_i||_\infty^2 < 1$, and $||\mu_i||_\infty$ is not a strictly convex norm. Also, the construction of the auxiliary cost function, $J_v(\beta)$, must now accomodate $n_0 + n_1$ terms. Finally, the translation of the cost function into those terms necessary for the application of optimal control theory may or may not be possible. Much work remains undone in this area, including the possible requirement of all new theory.

5.1.5 Sensitivity Issues

While we have talked about the sensitivity of the separating hyperplane to the value of ϵ, we have ignored other sensitivity issues. One is often interested in the sensitivity of the solution to perturbations of the system coefficient matrices A_i, B_i, and so on. If coefficients are slightly inaccurate, then we would like to say that the test function from the FDMI algorithm is still usable, but subject to some error. A sensitivity study identifies bounds on the error due to coefficient perturbations. It should be noted that, as in the small nonlinearities case discussed above, the algorithm can be modified to be robust to bounded perturbations of the system coefficients via the addition of more noise terms resulting in a higher energy than necessary detection signal. In that light, the development of some results in the sensitivity issue will closely follow those of the small nonlinearities problem. As an alternative, one could consider system matrices $A_i + \Delta A_i$, $B_i + \Delta B_i$, and so on, where ΔA_i and ΔB_i are small. System models for this approach are straightforward to write down, but will involve many more parameters than the unperturbed models. The optimal solution will depend on the perturbations, providing the sensitivity information one seeks, but solving the problem will be computationally expensive. Regardless of the approach taken, much more work on sensitivity issues remains.

5.2 Conclusions

Our goals for this thesis were to apply the multi-model approach to fault detection and model identification in linear descriptor systems, modeling noise as bounded energy signals, proving that this combination is a valid and efficient tool for these types of problems, and to develop an algorithm that demonstrates perfect fault detection and model identification. As we saw in Chapter 1, the combination of the multi-model approach and the bounded energy noise model is under-explored, in that very little existing literature reports their combined use. Most work in the multi-model approach uses statistical noise models which, while theoretically attractive, do not provide the computational friendliness to optimization software that the bounded energy noise models do. The remaining theory involves the single-model approach, utilizing either feedback or observer design for imperfect fault detection and model identification.

After laying the groundwork for the types of systems to be addressed and the numerical methods to be applied to those systems in Chapter 1, we developed the theory for the first half of the problem, fault detection, in Chapter 2. There, after defining the minimum energy detection signal, we constructed the auxiliary cost function. We then translated the problem into a nested optimization problem suitable for implementation in SOCS, solving for the necessary conditions for a minimum of the inner problem, and then using those conditions as constraints on the outer problem. We presented four forms of the MEDS algorithm, two of which apply matrix Riccati differential equation theory, which is well-suited to the limiting half-infinite interval case. After stating the algorithm, and the software in which we implemented it, we addressed several variations of the problem for which the algorithm is well suited, most notably the multiple fault model case, with which prior work in this area is not compatible.

Next we turned to the second half of the problem, model identification. Given an output, our detection signal ensured that only one model could have produced it. In Chapter 3 we developed the theory for the separating hyperplane for the output sets. We reduced the input of the noise to the system in order to translate the problem into an optimization problem, the solution of which gave an approximation to the normal of the separating hyperplane. The equation of the normal was implemented as a test function in the MI algorithm. After stating the algorithm, we again addressed variations of the problem, and showed that the second half of the algorithm is as equally suited to those variations as the first half.

Chapter 4 was dedicated to the presentation and analysis of various types of examples. We applied our implementation of the FDMI algorithm to one-, two-, and three-dimensional examples, demonstrating comparable performance on all. A multiple fault model case was examined, and the capability of the algorithm in this area was shown to be as expected. In fact, the only unexpected result came from an example that was constructed to force the SOCS version of the algorithm to fail. Instead of failing, the algorithm worked, demonstrating unanticipated robustness. Despite this success, some indications of difficulty were uncovered. Initial guess construction, starting mesh size, objective function scaling, and the selection of ϵ each showed signs of becoming factors in certain problems but not in others. These issues should be resolved before any real-world application of the algorithm.

Finally, in the present chapter, we introduced several extensions of the FDMI algorithm, describing the basics of each, but leaving the bulk of the work to future research. The tools developed in this thesis show much promise for numerically solving the fault detection and model identification problem in linear descriptor systems. Where previously there existed few multi-model methods for dealing with short decision horizons, this work provides a new approach for online fault detection and model identification.

List of References

[1] Athens, M., and P. L. Falb, *Optimal Control*, McGraw-Hill, 1966.

[2] Bennett, S. M., R. J. Patton, S. Daley, and D. A. Newton, Model Based Intermittent Fault Tolerance in an Induction Motor Drive, *Proceedings of the IMACS Multiconference CESA '96*, in the *Symposium on Control, Optimization and Supervision*, vol. 1, July 1996, pp. 678-683.

[3] Betts, J. T., M. J. Carter, and W. P. Huffman, *Software for Nonlinear Optimization*, Mathematics and Engineering Analysis, Library Report MEA-LR-083 R1, Boeing Information and Support Services, The Boeing Company, PO Box 3707, Seattle, WA 98124-2207, June 1997.

[4] Betts, J. T., and W. P. Huffman, *Sparse Optimal Control Software*, Mathematics and Engineering Analysis, Library Report MEA-LR-085, Boeing Information and Support Services, The Boeing Company, PO Box 3707, Seattle, WA 98124-2207, July 1997.

[5] Brenan, K. E., S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, 1996.

[6] Brogan, W. L., *Modern Control Theory*, Prentice-Hall, 1991.

[7] Campbell, S. L., A Survey of Time Varying and Nonlinear Descriptor Control Systems, *Proceedings of the Symposium on Implicit and Nonlinear Systems*, 1992, pp. 356-363.

[8] Campbell, S. L., High-Index Differential Algebraic Equations, *Mechanical Structures and Machines*, vol. 23, no. 2, 1995, pp. 199-222.

[9] Campbell, S. L., Linearization of DAEs along trajectories, *Z. angew Mathematical Physics*, vol. 46, 1995, pp. 70-84.

[10] Campbell, S. L., and E. Griepentrog, Solvability of General Differential Algebraic Equations, *SIAM Journal of Scientific Computing*, vol. 16, no. 2, March 1995, pp. 257-270.

[11] Campbell, S. L., K. Horton, R. Nikoukhah, and F. Delebecque, Rapid Model Selection and the Separability Index, *Proceedings of the 4th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS 2000)*, June 2000, pp. 1187-1192.

[12] Campbell, S. L., K. Horton, R. Nikoukhah, and F. Delebecque, Auxiliary signal design for rapid multi-model identification using optimization, preprint.

[13] Campbell, S. L., and E. Moore, Constraint preserving integrators for general nonlinear higher index DAEs, *Numerical Mathematics*, vol. 69, 1995, pp. 383-399.

[14] Campbell, S. L., and L. R. Petzold, Canonical Forms and Solvable Singular Systems of Differential Equations, *SIAM Journal of Algebraic Discretization Methods*, vol. 4, no. 4, December 1983, pp. 517-521.

[15] Chen, R. H., and J. L. Speyer, Residual-Sensitive Fault Detection Filter, preprint, January 1999.

[16] Chowdhury, F. N., and J. L. Aravena, A Modular Methodology for Fast Fault Detection and Classification in Power Systems, *IEEE Transactions on Control Systems Technology*, vol. 6, no. 5, September 1998, pp. 623-634.

[17] Chung, W. H., and J. L. Speyer, A Game Theoretic Fault Detection Filter, *IEEE Transactions on Automatic Control*, vol. 43, no. 2, February 1998, pp. 143-161.

[18] Heal, K. M., M. L. Hansen, and K. M. Rickard, *Maple V, Release 5 Learning Guide*, Waterloo Maple Inc, Ontario, Canada, 1998.

[19] Jonckheere, E. A., and G.-R. Yu, Propulsion Control of Crippled Aircraft by H_∞ Model Matching, *IEEE Transactions on Control Systems Technology*, vol. 7, no. 2, March 1999, pp. 142-159.

[20] Kalman, R. E., et al., Fundamental Study of Adaptive Control Systems, *Wright-Patterson Air Force Base Tech. Rept.* ASD-TR-61-27, vol. 1, April, 1962.

[21] Keller, J. Y., L. Summerer, and M. Darouach, Robust Failure Detection from a Multi-Model Approach, *Proceedings of the IMACS Multiconference CESA '96*, in the *Symposium on Control, Optimization and Supervision*, vol. 1, July 1996, pp. 384-388.

[22] Kerestecioğlu, F., *Change Detection and Input Design in Dynamical Systems*, Research Studies Press, Taunton, U.K., 1993.

[23] Kerestecioğlu, F., and M. B. Zarrop, Input design for detection of abrupt changes in dynamical systems, *International Journal of Control*, vol. 59, 1994, pp. 1063-1084.

[24] Koenig, D., S. Nowakowski, and T. Cecchin, A Comparative Study of Unknown

Input Observers Design Methods Applied for Fault Detection, Isolation, and Correction, *Proceedings of the IMACS Multiconference CESA '96*, in the *Symposium on Control, Optimization and Supervision*, vol. 1, July 1996, pp. 665-671.

[25] Luenberger, D. G., *Optimization by Vector Space Methods*, John Wiley and Sons, 1969.

[26] The MathWorks, Inc, *Using MATLAB Version 5*, 1998.

[27] Nagpal, K.M., and P. P. Khargonekar, Filtering and Smoothing in an H^∞ Setting, *IEEE Transactions on Automatic Control*, vol. 36, no. 2, February 1991. pp. 152-166.

[28] Nikoukhah, R., Guaranteed Active Failure Detection and Isolation for Linear Dynamical Systems, *Automatica*, vol. 34, no. 11, 1998, pp. 1345-1358.

[29] Nikoukhah, R., S. L. Campbell, and F. Delebecque, Detection signal design for failure detection: a robust approach, *International Journal of Adaptive Control Signal Processes* 2000, vol. 14, pp. 701-724.

[30] Nikoukhah, R., S. L. Campbell, Kirk Horton, and F. Delebecque, Auxiliary signal design for robust multi-model identification, *IEEE Transactions on Automatic Control*, accepted subject to final revision.

[31] Nikoukhah, R., F. Delebecque, S. L. Campbell, and K. Horton, Multi-model Identification and the Separability Index, *Proceedings of the 14th International Symposium of the Mathematical Theory of Networks and Systems 2000*, June 2000, CDROM.

[32] Nuninger, W., F. Kratz, and J. Ragot, Observers and Redundancy Equations

Generation for Systems with Unknown Inputs, *Proceedings of the IMACS Multiconference CESA '96*, in the *Symposium on Control, Optimization and Supervision*, vol. 1, July 1996, pp. 672-677.

[33] Riggins, R. N., and W. B. Ribbens, Designed Inputs for Detection and Isolation of Failures in the State Transition Matrices of Dynamic Systems, *IEEE Transactions on Control Systems Technology*, vol. 5, no. 2, March 1997, pp. 149-161.

[34] Rockafellar, R. T., *Convex Analysis*, Princeton University Press, 1970.

[35] Sauter, D., H. Noura, F. Hamelin, and D. Theilliol, Parity Space Approach for Fault Diagnosis in Descriptor Systems, *Proceedings of the IMACS Multiconference CESA '96*, in the *Symposium on Control, Optimization and Supervision*, vol. 1, July 1996, pp. 380-383.

[36] Tadmor, G., Worst-Case Design in the Time Domain: The Maximum Principle and the Standard H_∞ Problem, *Mathematical Control Signals Systems*, vol. 3, 1990, pp. 301-324.

[37] Uosaki, K., I. Tanaka, and H. Sugiyama, Optimal input design for autoregressive model discrimination with constrained output variance, *IEEE Transactions on Automatic Control*, AC-29, 1984, pp. 348-350.

[38] Vincent, T. L., and W. J. Grantham, *Nonlinear and Optimal Control Systems*, John Wiley and Sons, 1997.

[39] Yang, J.-S., Mixed H^2 Compensator Design for an Aircraft Control Problem, *Proceedings of the 38th Conference on Decision and Control*, December 1999, pp. 1964-1969.

[40] Youssouf, A., and M. Kinnaert, Residual Generation for Singular Systems using Parity Relations, *Proceedings of the IMACS Multiconference CESA '96*, in the

Symposium on Control, Optimization and Supervision, vol. 1, July 1996, pp. 533-538.

[41] Zhang, X. J., *Auxiliary Signal Design in Fault Detection and Diagnosis*, Springer, Heidelberg, 1989.

[42] Zhang, Y., and J. Jiang, An Interacting Multiple-Model Based Fault Detection, Diagnosis and Fault-Tolerant Control Approach, *Proceedings of the 38th Conference on Decision and Control*, December 1999, pp. 3593-3598.

[43] Zhang, Y., and J. Jiang, Design of Integrated Fault Detection, Diagnosis and Reconfigurable Control Systems, *Proceedings of the 38th Conference on Decision and Control*, December 1999, pp. 3587-3592.

Appendix A

Software Drivers

Several pieces of commercial software can be combined to implement the FDMI algorithm presented in this thesis. Model reduction is efficiently accomplished in MATLAB, by The MathWorks, Inc. [26]. FORTRAN code generation of the reduced system model is done in MAPLE, by Waterloo Maple, Inc. [18]. Optimization is carried out in SOCS, by Boeing [3, 4] for both the minimum energy detection signal and the normal to the separating hyperplane. MATLAB is used to analyze the output from SOCS and create plots. Sample driver files for each phase of this process are included below.

A.1 Model Reduction

The transformation of the original system to the reduced model discussed in Chapter 2 is accomplished by the following MATLAB m-file.

```
%reduced2d.m
%current parameters for: common mode example (exam10)

%system matrices
A0=[1 0;
    0 2];
A1=[1 0;
```

```
    0 3];
B0=[2 1;
    1 2];
B1=[2 1;
    1 2];
C0=[1 0;
    0 1];
C1=[1 0;
    0 1];
M0=[0 0 0 1;
    0 0 1 0 ];
M1=[0 0 0 1;
    0 0 1 0];
N0=[0 1 0 0;
    1 0 0 0];
N1=[0 1 0 0;
    1 0 0 0];

%may need to do constant orthogonal change of coords on the noise
%do it via a QR decomp on N_i^T to get [Nb_i  0],
%where Nb_i is invertible, also gives [Mb_i  Mt_i]
[Q0,R0]=qr(N0')
[Q1,R1]=qr(N1')
pause

%now N_i^T = Q_i * R_i, so N_i = R_i^T * Q_i^T
%and Q_i^T is an orthogonal matrix
%absorb Q_i^T into the noise vector nu_i to get new noise vector
%and Nb_i becomes the invertible part of R_i^T

%may need to fix signs in Q_i and/or R_i
R0=-R0;
R1=-R1;
Q0(1,1)=-Q0(1,1);
Q0(2,2)=-Q0(2,2);
Q1(1,1)=-Q1(1,1);
Q1(2,2)=-Q1(2,2);

%break down into Mb_i, Mt_i, Nb_i, and Nt_i (zeros)
[mN0,nN0]=size(N0);
[mN1,nN1]=size(N1);
mnN0=min(mN0,nN0);
```

```
mnN1=min(mN1,nN1);
mxN0=max(mN0,nN0);
mxN1=max(mN1,nN1);
N0n=R0';
N1n=R1';
Nb0=N0n(1:mnN0,1:mnN0);
Nb1=N1n(1:mnN1,1:mnN1);
Nt0=N0n(:,mnN0+1:mxN0);  %just need the size of this
Nt1=N1n(:,mnN1+1:mxN1);  %and this
M0n=M0*Q0';
M1n=M1*Q1';
Mb0=M0n(:,1:mnN0);
Mb1=M1n(:,1:mnN1);
Mt0=M0n(:,mnN0+1:mxN0);
Mt1=M1n(:,mnN1+1:mxN1);

%create reduced model system matrices
[mA0,nA0]=size(A0);
[mA1,nA1]=size(A1);
[mMt0,nMt0]=size(Mt0);
[mMt1,nMt1]=size(Mt1);
[mNb,nNb]=size(Nb1);
[mNt0,nNt0]=size(Nt0);
[mNt1,nNt1]=size(Nt1);

A=[A0-Mb0*inv(Nb0)*C0 Mb0*inv(Nb0)*C1; zeros(mA1,nA0) A1]
M=[Mt0 Mb0*inv(Nb0)*Nb1 zeros(mMt0,nMt1); zeros(mMt1,nMt0) Mb1 Mt1]
B=[B0;B1]
C=[inv(Nb0)*C0 -inv(Nb0)*C1];
N=inv(Nb0)*[zeros(mNb,nNt0) Nb1 zeros(mNb,nNt1)];

%Q and H without beta
Qnb=2*C'*C
Hnb=-4*C'*N

%size of upper left block of R
ulident=nNt0

%Nb1'Nb0'^-1Nb0^-1Nb1 for the center block of R
Nb1
Nmess=Nb1'*inv(Nb0')*inv(Nb0)*Nb1
```

```
%size of lower right block of R
lrident=nNt1

%end of routine
```

Outputs from this routine are used as follows:

- sizes of reduced model system matrices are inputted into MAPLE,
- parameters of the R matrix are inputted into MAPLE,
- parameters of the reduced model system matrices are inputted into SOCS.

A.2 Fortran Code Generation

The following MAPLE routine takes the sizes of reduced model system matrices and the parameters of the R matrix and creates FORTRAN code for pasting into the SOCS FORTRAN driver.

```
%reduced2d.mws
> restart;
> with(linalg):

%sizes of reduced system matrices
> A:=matrix(4,4);
> B:=matrix(4,1);
> M:=matrix(4,5);
> Q:=matrix(4,4);
> H:=matrix(4,5);
> Nmess:=matrix(1,1);

%structure of the R matrix
> R:=matrix(5,5,[2*P[1],0,0,0,0,0,2*P[1],0,0,0,0,0,2*(1-P[1])+
2*P[1]*Nmess[1,1],0,0,0,0,0,2*(1-P[1]),0,0,0,0,0,2*(1-P[1])]);

%SOCS needs all variables in a single vector
> Z:=vector(15);
> x:=vector(4,[Z[1],Z[2],Z[3],Z[4]]);
> lambda:=vector(4,[Z[5],Z[6],Z[7],Z[8]]);
> nu:=vector(5,[Z[10],Z[11],Z[12],Z[13],Z[14]]);
> v:=vector(1,[Z[15]]);

%create constraint equations
> F1:=matadd(matadd(multiply(A,x),multiply(B,v)),multiply(M,nu));
> F2:=matadd(matadd(multiply(-P[1]*Q,x),multiply(-P[1]*H,nu)/2),
multiply(transpose(-A),lambda));
> F3:=(multiply(multiply(transpose(x),P[1]*Q),x)
+multiply(multiply(transpose(x),P[1]*H),nu)
+multiply(multiply(transpose(nu),R),nu))/2;
> F4:=matadd(matadd(multiply(R,nu),
multiply(transpose(P[1]*H),x)/2),multiply(transpose(M),lambda));

%translate constraint equations into FORTRAN code
> fortran(F1);
> fortran(F2);
> fortran(F3);
> fortran(F4);

%end of routine
```

A.3 Optimization via the FDMI Algorithm

The following SOCS driver takes the FORTRAN code output by the above MAPLE routine, as well as the parameters of the reduced model system matrices from the above MATLAB routine, to accomplish both halves of the FDMI algorithm.

```
C Main driver: 2dex4m.f

      PROGRAM MID42D

C Generic 2D FDMI problem on [T0,TF] in formulation 2nR
C using min distance between convex set theory

C Need to set problem parameters in seven places:
C a few lines down from here, and in OPT2IN, MID2IN (T0,TF only)
C near end of MAIN (output file names in open calls)
C in OPT2DE (all coordinate changed parameters)
C in MID2DE (all original problem parameters plus epsilon (EPSIL))
C in function YBAR(T) (coordinate changed parameters C1 and NB1)
```

```
C current parameters are for: multimod2d_12

      INTEGER I,NIW,MAXPHS,NW,MAXCS,MAXDP

      PARAMETER (NIW = 10000000,MAXPHS = 5,NW = 10000000,MAXCS=100000)
      PARAMETER (MAXDP = 1000)
      INTEGER IW(NIW),IPCPH(MAXPHS+1),IPDPH(MAXPHS+1),NEEDED,IER
      DOUBLE PRECISION T0,TF,T,W(NW),CSTAT(MAXCS),DPARM(MAXDP),YBAR
      PARAMETER (T0=0.D0,TF=1.D0)

C OCSEVL will be called during second SOCS run (MI) to extract
C output from first SOCS run (MEDS) needed in MI optimization, so need
C to save some of the parameters from first SOCS run that are args
C to OCSEVL to new variable names as defined below

      INTEGER IPCPH2(MAXPHS+1),IPDPH2(MAXPHS+1)
      DOUBLE PRECISION CSTAT2(MAXCS),DPARM2(MAXDP),W2(NW)

C Also need them to be common because SOCS will not pass them
C to the functions that use them

      COMMON CSTAT2,IPCPH2,DPARM2,IPDPH2,W2

      INTEGER IPHASE,NDP,IOUNIT,IOFLAG,NPTAUX,MXSTAT,NDYN
      PARAMETER (MXSTAT = 20)
      DOUBLE PRECISION STSKL(0:MXSTAT),DTAUX

      EXTERNAL OPT2IN,MID2IN,DUMYIG,OPT2DE,MID2DE
      EXTERNAL DUMYPF,DUMYPR

C Output verbosity
CC      CALL HHSOCS('IPGRD=0')
C  default=10
C      CALL HHSOCS('IPFSFD=0') default=0
CC      CALL HHSOCS('IPNLP=0')
C  default=10
C      CALL HHSOCS('IPODE=0')  default=0
C      CALL HHSNLP('IOFLAG=0')
C  default=10
      CALL HHSNLP('MAXNFE=50000')
      CALL HHSNLP('NITMAX=1000')
      CALL HHSNLP('KTOPTN=SMALL')
      CALL HHSNLP('ALGOPT=FM')
      CALL HHSOCS('MITODE=15')
      CALL HHSOCS('ODETOL=1.D-7')
      CALL HHSOCS('SPRTHS=SPARSE')
      CALL HHSOCS('MTSWCH=3')
      CALL HHSOCS('NSSWCH=1')
C Optional tolerances for quicker convergence
C      CALL HHSOCS('ODETOL=0.2D-4')
C      CALL HHSNLP('CONTOL=1.D-5')
C      CALL HHSNLP('OBJTOL=1.D-5')

      WRITE(*,*) 'MADE IT: START OF MIN V PROBLEM'

C OPTIMIZATION TO FIND MIN ENERGY PROPER V - first SOCS call

      CALL HDSOCS(OPT2IN,DUMYIG,OPT2DE,DUMYPF,DUMYPR,
     +            IW,NIW,W,NW,MAXPHS,CSTAT,MAXCS,IPCPH,DPARM,MAXDP,
     +            IPDPH,NEEDED,IER)

C Transfer args for OCSEVL

      DO 10 I = 1,MAXCS
       CSTAT2(I) = CSTAT(I)
 10   CONTINUE
      DO 20 I = 1,MAXPHS+1
       IPCPH2(I) = IPCPH(I)
       IPDPH2(I) = IPDPH(I)
 20   CONTINUE
      DO 30 I = 1,MAXDP
       DPARM2(I) = DPARM(I)
 30   CONTINUE
      DO 40 I = 1,NW
       W2(I) = W(I)
 40   CONTINUE

C Output YBAR and optimal solution from MEDS phase

      IOUNIT=32
      OPEN(IOUNIT,FILE='thrmod2d_1yb.dat',STATUS='UNKNOWN')
      DO 50 I = 1,251
      T = T0+(TF-T0)*(I-1)/2.5D2
      WRITE(IOUNIT,*) YBAR1(T),YBAR2(T),YBAR3(T)
 50   CONTINUE
      CLOSE(IOUNIT)

      OPEN(IOUNIT,FILE='thrmod2d_1ds.dat',STATUS='UNKNOWN')
      NPTAUX = 250
      NDP = 15
      NDYN = 16
      DTAUX = 0
      IOFLAG = 10
      IPHASE = 1
      CALL DFILL(NDYN,1.D0,STSKL,1)
      CALL AUXOUT(IPHASE,CSTAT,DPARM,NDP,STSKL,W,NW,IOUNIT,
     +                 IOFLAG,NPTAUX,DTAUX,DUMYPR)
      CLOSE(IOUNIT)

C OPTIMIZATION TO FIND TEST FUNCTION - second SOCS call

      WRITE(*,*) 'MADE IT: START OF TEST FN PROBLEM'

      CALL HDSOCS(MID2IN,DUMYIG,MID2DE,DUMYPF,DUMYPR,
     +            IW,NIW,W,NW,MAXPHS,CSTAT,MAXCS,IPCPH,DPARM,MAXDP,
     +            IPDPH,NEEDED,IER)

C Output complete optimal solution from MI phase

      IOUNIT = 32
      OPEN(IOUNIT,FILE='thrmod2d_1m7.dat',STATUS='UNKNOWN')
      NPTAUX = 250
      NDP = 14
      NDYN = 15
      DTAUX = 0
      IOFLAG = 10
      IPHASE = 1
      CALL DFILL(NDYN,1.D0,STSKL,1)
      CALL AUXOUT(IPHASE,CSTAT,DPARM,NDP,STSKL,W,NW,IOUNIT,
     +                 IOFLAG,NPTAUX,DTAUX,DUMYPR)
      CLOSE(IOUNIT)

      END
```

```
C MEDS phase driver

      SUBROUTINE OPT2IN(IPHASE,NPHS,METHOD,NSTG,NCF,NPF,NPV,NAV,
     +                  NGRID,INIT,MAXMIN,MXPARM,P0,PLB,PUB,PLBL,
     +                  MXSTAT,Y0,Y1,YLB,YUB,STSKL,STLBL,MXPCON,CLB,
     +                  CUB,CLBL,MXTERM,COEF,ITERM,TITLE,IER)

      INTEGER IPHASE,NPHS,METHOD,NSTG,NCF(3),NPF(2),NPV,NAV,NGRID,
     +        INIT,MAXMIN,MXPARM,MXSTAT,MXPCON,MXTERM,ITERM(4,MXTERM),
     +        IER

      DOUBLE PRECISION P0(MXPARM),PLB(MXPARM),PUB(MXPARM),Y0(0:MXSTAT),
     +          Y1(0:MXSTAT),YLB(-1:1,0:MXSTAT),YUB(-1:1,0:MXSTAT),
     +          STSKL(0:MXSTAT+MXPARM,2),CLB(MXPCON),CUB(MXPCON),
     +          COEF(MXTERM)

      CHARACTER PLBL(MXPARM+2)*80,STLBL(0:MXSTAT)*80,
     +          CLBL(0:MXPCON)*80,TITLE(3)*60

      DOUBLE PRECISION T0,TF
      PARAMETER (T0=0.D0,TF=1.D0)

      METHOD = 3
      NSTG = 1
      NCF(1) = 9
      NCF(2) = 5
      NCF(3) = 1
      NPF(2) = 0
      NAV = 6
      NPV = 1
      NGRID = 11
      INIT = 1
      MAXMIN = -1

C Initial and final time

      Y0(0) = T0
      Y1(0) = TF

C Initial guesses, v = 1, beta somewhere in the interior of [0,1]

      DO 10 I = 1,14
```

```
           Y0(I) = 0.D0
 10     CONTINUE
        Y0(15) = 1.D0

        DO 20 I = 1,14
           Y1(I) = 0.D0
 20     CONTINUE
        Y1(15) = 1.D0

        P0(1) = 0.2D0

C Fix the boundary conditions

        DO 30 I = 5,9
        YLB(-1,I) = 0.D0
        YUB(-1,I) = 0.D0
 30     CONTINUE

        DO 40 I = 5,8
        YLB(1,I) = 0.D0
        YUB(1,I) = 0.D0
 40     CONTINUE
        YLB(1,9) = 1.D0

C Bound BETA (avoiding singularities at 0 and 1)

        PLB(1) = 0.01D0
        PUB(1) = 0.99D0

C Fix the start and finish time

        YLB(-1,0) = Y0(0)
        YUB(-1,0) = Y0(0)
        YLB(1,0) = Y1(0)
        YUB(1,0) = Y1(0)

C Equality constraints

        DO 50 I = 1,5
         ITERM(1,I) = I
         ITERM(2,I) = 1
         ITERM(3,I) = 0
```

```
       ITERM(4,I) = -I
       CLB(I) = 0.D0
       CUB(I) = 0.D0
 50   CONTINUE

C Objective function

      ITERM(1,6) = 0
      ITERM(2,6) = 1
      ITERM(3,6) = 0
      ITERM(4,6) = -6

      RETURN
      END


C MEDS constraint definition

      SUBROUTINE OPT2DE(IPHASE,T,Z,NZ,P,NP,F,NF,IFERR)

      INTEGER IPHASE,NZ,NP,NF,IFERR
      DOUBLE PRECISION T,Z(NZ),P(NP),F(NF)
      DOUBLE PRECISION A(4,4),B(4,1),S1,S2
      DOUBLE PRECISION M(4,5),H(4,5),Q(4,4),Nmess(1,1)

C Nine ODE constraints
C Variable list: x:Z(1)-Z(4), lambda:Z(5)-Z(8), Z:Z(9),
C                nu:Z(10)-Z(14), v:Z(15), beta:P(1)

      INTEGER R,C

C Problem parameters

      DATA ((A(R,C),C=1,4),R=1,4)/-1.D0,1.D0,0.D0,0.D0,
     #-1.D0,-3.D0,0.D0,0.D0,
     #0.D0,0.D0,-10.D0,1.D0,
     #0.D0,0.D0,-1.D0,-3.D0/
      DATA ((B(R,C),C=1,1),R=1,4)/1.D0,0.D0,1.D0,0.D0/
      DATA ((M(R,C),C=1,5),R=1,4)/0.D0,1.D0,0.D0,0.D0,0.D0,
     #1.D0,0.D0,0.D0,0.D0,0.D0,
     #0.D0,0.D0,0.D0,0.D0,1.D0,
     #0.D0,0.D0,0.D0,1.D0,0.D0/
```

```
      DATA ((Q(R,C),C=1,4),R=1,4)/2.D0,2.D0,-2.D0,-2.D0,
     #2.D0,2.D0,-2.D0,-2.D0,
     #-2.D0,-2.D0,2.D0,2.D0,
     #-2.D0,-2.D0,2.D0,2.D0/
      DATA ((H(R,C),C=1,5),R=1,4)/0.D0,0.D0,-4.D0,0.D0,0.D0,
     #0.D0,0.D0,-4.D0,0.D0,0.D0,
     #0.D0,0.D0,4.D0,0.D0,0.D0,
     #0.D0,0.D0,4.D0,0.D0,0.D0/
      DATA ((Nmess(R,C),C=1,1),R=1,1)/1.D0/

C x'
      F(1) = A(1,1)*Z(1)+A(1,2)*Z(2)+A(1,3)*Z(3)+A(1,4)*Z(4)+B(1,1)*Z(1
     #5)+M(1,1)*Z(10)+M(1,2)*Z(11)+M(1,3)*Z(12)+M(1,4)*Z(13)+M(1,5)*Z(14
     #)
      F(2) = A(2,1)*Z(1)+A(2,2)*Z(2)+A(2,3)*Z(3)+A(2,4)*Z(4)+B(2,1)*Z(1
     #5)+M(2,1)*Z(10)+M(2,2)*Z(11)+M(2,3)*Z(12)+M(2,4)*Z(13)+M(2,5)*Z(14
     #)
      F(3) = A(3,1)*Z(1)+A(3,2)*Z(2)+A(3,3)*Z(3)+A(3,4)*Z(4)+B(3,1)*Z(1
     #5)+M(3,1)*Z(10)+M(3,2)*Z(11)+M(3,3)*Z(12)+M(3,4)*Z(13)+M(3,5)*Z(14
     #)
      F(4) = A(4,1)*Z(1)+A(4,2)*Z(2)+A(4,3)*Z(3)+A(4,4)*Z(4)+B(4,1)*Z(1
     #5)+M(4,1)*Z(10)+M(4,2)*Z(11)+M(4,3)*Z(12)+M(4,4)*Z(13)+M(4,5)*Z(14
     #)

C lambda'
      F(5) = -P(1)*Q(1,1)*Z(1)-P(1)*Q(1,2)*Z(2)-P(1)*Q(1,3)*Z(3)-P(1)*Q
     #(1,4)*Z(4)-P(1)*H(1,1)*Z(10)/2-P(1)*H(1,2)*Z(11)/2-P(1)*H(1,3)*Z(1
     #2)/2-P(1)*H(1,4)*Z(13)/2-P(1)*H(1,5)*Z(14)/2-A(1,1)*Z(5)-A(2,1)*Z(
     #6)-A(3,1)*Z(7)-A(4,1)*Z(8)
      F(6) = -P(1)*Q(2,1)*Z(1)-P(1)*Q(2,2)*Z(2)-P(1)*Q(2,3)*Z(3)-P(1)*Q
     #(2,4)*Z(4)-P(1)*H(2,1)*Z(10)/2-P(1)*H(2,2)*Z(11)/2-P(1)*H(2,3)*Z(1
     #2)/2-P(1)*H(2,4)*Z(13)/2-P(1)*H(2,5)*Z(14)/2-A(1,2)*Z(5)-A(2,2)*Z(
     #6)-A(3,2)*Z(7)-A(4,2)*Z(8)
      F(7) = -P(1)*Q(3,1)*Z(1)-P(1)*Q(3,2)*Z(2)-P(1)*Q(3,3)*Z(3)-P(1)*Q
     #(3,4)*Z(4)-P(1)*H(3,1)*Z(10)/2-P(1)*H(3,2)*Z(11)/2-P(1)*H(3,3)*Z(1
     #2)/2-P(1)*H(3,4)*Z(13)/2-P(1)*H(3,5)*Z(14)/2-A(1,3)*Z(5)-A(2,3)*Z(
     #6)-A(3,3)*Z(7)-A(4,3)*Z(8)
      F(8) = -P(1)*Q(4,1)*Z(1)-P(1)*Q(4,2)*Z(2)-P(1)*Q(4,3)*Z(3)-P(1)*Q
     #(4,4)*Z(4)-P(1)*H(4,1)*Z(10)/2-P(1)*H(4,2)*Z(11)/2-P(1)*H(4,3)*Z(1
     #2)/2-P(1)*H(4,4)*Z(13)/2-P(1)*H(4,5)*Z(14)/2-A(1,4)*Z(5)-A(2,4)*Z(
     #6)-A(3,4)*Z(7)-A(4,4)*Z(8)
```

```
C Z’
      S2 = (P(1)*Q(1,1)*Z(1)+Z(2)*P(1)*Q(2,1)+Z(3)*P(1)*Q(3,1)+Z(4)*P(1)
     #*Q(4,1))*Z(1)/2+(Z(1)*P(1)*Q(1,2)+P(1)*Q(2,2)*Z(2)+Z(3)*P(1)*Q(3,2
     #)+Z(4)*P(1)*Q(4,2))*Z(2)/2+(Z(1)*P(1)*Q(1,3)+Z(2)*P(1)*Q(2,3)+P(1)
     #*Q(3,3)*Z(3)+Z(4)*P(1)*Q(4,3))*Z(3)/2
      S1 = s2+(Z(1)*P(1)*Q(1,4)+Z(2)*P(1)*Q(2,4)+Z(3)*P(1)*Q(3,4)+P(1)*Q
     #(4,4)*Z(4))*Z(4)/2+(Z(1)*P(1)*H(1,1)+Z(2)*P(1)*H(2,1)+Z(3)*P(1)*H(
     #3,1)+Z(4)*P(1)*H(4,1))*Z(10)/2+(Z(1)*P(1)*H(1,2)+Z(2)*P(1)*H(2,2)+
     #Z(3)*P(1)*H(3,2)+Z(4)*P(1)*H(4,2))*Z(11)/2+(Z(1)*P(1)*H(1,3)+Z(2)*
     #P(1)*H(2,3)+Z(3)*P(1)*H(3,3)+Z(4)*P(1)*H(4,3))*Z(12)/2
      F(9)=s1+(Z(1)*P(1)*H(1,4)+Z(2)*P(1)*H(2,4)+Z(3)*P(1)*H(3,4)+Z(4)*P
     #(1)*H(4,4))*Z(13)/2+(Z(1)*P(1)*H(1,5)+Z(2)*P(1)*H(2,5)+Z(3)*P(1)*H
     #(3,5)+Z(4)*P(1)*H(4,5))*Z(14)/2+Z(10)**2*P(1)+Z(11)**2*P(1)+Z(12)*
     #*2*(2-2*P(1)+2*P(1)*Nmess(1,1))/2 +Z(13)**2*(2-2*P(1))/2+Z(14)**2*
     #(2-2*P(1))/2

C Five algebraic constraints
      F(10) = 2*Z(10)*P(1)+Z(1)*P(1)*H(1,1)/2+Z(2)*P(1)*H(2,1)/2+Z(3)*P(
     #1)*H(3,1)/2+Z(4)*P(1)*H(4,1)/2+M(1,1)*Z(5)+M(2,1)*Z(6)+M(3,1)*Z(7)
     #+M(4,1)*Z(8)
      F(11) = 2*Z(11)*P(1)+Z(1)*P(1)*H(1,2)/2+Z(2)*P(1)*H(2,2)/2+Z(3)*P(
     #1)*H(3,2)/2+Z(4)*P(1)*H(4,2)/2+M(1,2)*Z(5)+M(2,2)*Z(6)+M(3,2)*Z(7)
     #+M(4,2)*Z(8)
      F(12) = Z(12)*(2-2*P(1)+2*P(1)*Nmess(1,1)) +Z(1)*P(1)*H(1,3)/2+Z(2
     #)*P(1)*H(2,3)/2+Z(3)*P(1)*H(3,3)/2+Z(4)*P(1)*H(4,3)/2+M(1,3)*Z(5)+
     #M(2,3)*Z(6)+M(3,3)*Z(7)+M(4,3)*Z(8)
      F(13) = Z(13)*(2-2*P(1))+Z(1)*P(1)*H(1,4)/2+Z(2)*P(1)*H(2,4)/2+Z(3
     #)*P(1)*H(3,4)/2+Z(4)*P(1)*H(4,4)/2+M(1,4)*Z(5)+M(2,4)*Z(6)+M(3,4)*
     #Z(7)+M(4,4)*Z(8)
      F(14) = Z(14)*(2-2*P(1))+Z(1)*P(1)*H(1,5)/2+Z(2)*P(1)*H(2,5)/2+Z(3
     #)*P(1)*H(3,5)/2+Z(4)*P(1)*H(4,5)/2+M(1,5)*Z(5)+M(2,5)*Z(6)+M(3,5)*
     #Z(7)+M(4,5)*Z(8)

C Objective function is a quadrature
      F(15) = Z(15)**2

      RETURN
      END


C MI phase driver
```

```
      SUBROUTINE MID2IN(IPHASE,NPHS,METHOD,NSTG,NCF,NPF,NPV,NAV,
     +                  NGRID,INIT,MAXMIN,MXPARM,P0,PLB,PUB,PLBL,
     +                  MXSTAT,Y0,Y1,YLB,YUB,STSKL,STLBL,MXPCON,CLB,
     +                  CUB,CLBL,MXTERM,COEF,ITERM,TITLE,IER)

      INTEGER IPHASE,NPHS,METHOD,NSTG,NCF(3),NPF(2),NPV,NAV,NGRID,
     +        INIT,MAXMIN,MXPARM,MXSTAT,MXPCON,MXTERM,ITERM(4,MXTERM),
     +        IER

      DOUBLE PRECISION P0(MXPARM),PLB(MXPARM),PUB(MXPARM),Y0(0:MXSTAT),
     +          Y1(0:MXSTAT),YLB(-1:1,0:MXSTAT),YUB(-1:1,0:MXSTAT),
     +          STSKL(0:MXSTAT+MXPARM,2),CLB(MXPCON),CUB(MXPCON),
     +          COEF(MXTERM)

      CHARACTER PLBL(MXPARM+2)*80,STLBL(0:MXSTAT)*80,
     +          CLBL(0:MXPCON)*80,TITLE(3)*60

      DOUBLE PRECISION T0,TF
      PARAMETER (T0=0.D0,TF=1.D0)

      METHOD = 3
      NSTG = 1
      NCF(1) = 6
      NCF(2) = 2
      NCF(3) = 1
      NPF(2) = 0
      NAV = 8
      NPV = 0
      NGRID = 11
      INIT = 1
      MAXMIN = -1

C Initial and final time

      Y0(0) = T0
      Y1(0) = TF

C Initial guesses

      DO 10 I = 1,13
         Y0(I) = 1.D0
 10   CONTINUE
```

```
      Y0(14)=0.D0

      DO 20 I = 1,13
         Y1(I) = 0.D0
 20   CONTINUE
      Y1(14)=1.D0

C Boundary conditions

      DO 30 I = 5,6
       YLB(-1,I) = 0.D0
       YUB(-1,I) = 0.D0
 30   CONTINUE
      DO 40 I = 5,6
       YUB(1,I) = 1.D0
 40   CONTINUE

C Fix the start and finish time

      YLB(-1,0) = Y0(0)
      YUB(-1,0) = Y0(0)
      YLB(1,0) = Y1(0)
      YUB(1,0) = Y1(0)

C Equality constraints

      ITERM(1,1) = 1
      ITERM(2,1) = 1
      ITERM(3,1) = 0
      ITERM(4,1) = -1
      CLB(1) = 0.D0
      CUB(1) = 0.D0

      ITERM(1,2) = 2
      ITERM(2,2) = 1
      ITERM(3,2) = 0
      ITERM(4,2) = -2
      CLB(2) = 0.D0
      CUB(2) = 0.D0

C Objective function
```

```
      ITERM(1,3) = 0
      ITERM(2,3) = 1
      ITERM(3,3) = 0
      ITERM(4,3) = -3

      RETURN
      END


C MI constraint definition

      SUBROUTINE MID2DE(IPHASE,T,Z,NZ,P,NP,F,NF,IFERR)

      INTEGER IPHASE,NZ,NP,NF,IFERR
      DOUBLE PRECISION T,Z(NZ),P(NP),F(NF),VH,VHAT,EPSIL
      DOUBLE PRECISION A0(2,2),A1(2,2),B0(2),B1(2),C0(2),C1(2)
      DOUBLE PRECISION M0(2,3),M1(2,3),N0(3),N1(3)

C Six ODE constraints
C Variable list: x:Z(1)-Z(4), q:Z(5)-Z(6), nu:Z(7)-Z(12), y:Z(13)-Z(14)

      PARAMETER (EPSIL=0.7D0)

      INTEGER R,C

C Problem parameters

      DATA ((A0(R,C),C=1,2),R=1,2)/-1.D0,1.D0,-1.D0,-3.D0/
      DATA ((A1(R,C),C=1,2),R=1,2)/-10.D0,1.D0,-1.D0,-3.D0/
      DATA B0/1.D0,0.D0/,B1/1.D0,0.D0/,C0/1.D0,1.D0/,C1/1.D0,1.D0/
      DATA ((M0(R,C),C=1,3),R=1,2)/0.D0,0.D0,1.D0,0.D0,1.D0,0.D0/
      DATA N0/1.D0,0.D0,0.D0/
      DATA ((M1(R,C),C=1,3),R=1,2)/0.D0,0.D0,1.D0,0.D0,1.D0,0.D0/
      DATA N1/1.D0,0.D0,0.D0/

C First compute VH

      VH = VHAT(T)

C x'
      F(1) = A0(1,1)*Z(1)+A0(1,2)*Z(2)+B0(1)*VH+EPSIL*(M0(1,1)*Z(7)
     #+M0(1,2)*Z(8)+M0(1,3)*Z(9))
```

```
      F(2) = A0(2,1)*Z(1)+A0(2,2)*Z(2)+B0(2)*VH+EPSIL*(M0(2,1)*Z(7)
     #+M0(2,2)*Z(8)+M0(2,3)*Z(9))
      F(3) = A1(1,1)*Z(3)+A1(1,2)*Z(4)+B1(1)*VH+EPSIL*(M1(1,1)*Z(10)
     #+M1(1,2)*Z(11)+M1(1,3)*Z(12))
      F(4) = A1(2,1)*Z(3)+A1(2,2)*Z(4)+B1(2)*VH+EPSIL*(M1(2,1)*Z(10)
     #+M1(2,2)*Z(11)+M1(2,3)*Z(12))

C q'
      F(5) = Z(7)**2+ Z(8)**2+ Z(9)**2
      F(6) = Z(10)**2+Z(11)**2+Z(12)**2

C Two algebraic constraints
      F(7) = C0(1)*Z(1)+C0(2)*Z(2)+EPSIL*(N0(1)*Z(7)+N0(2)*Z(8)
     #+N0(3)*Z(9))-Z(13)
      F(8) = C1(1)*Z(3)+C1(2)*Z(4)+EPSIL*(N1(1)*Z(10)+N1(2)*Z(11)
     #+N1(3)*Z(12))-Z(14)

C Obj function is a quadrature
      F(9) = (Z(13)-Z(14))**2

      RETURN
      END


C Function to extract MEDS solution VHAT for MI phase

      DOUBLE PRECISION FUNCTION VHAT(T)

      INTEGER MAXPHS,NW,MAXCS,MAXDP,IPHASE,IER
      PARAMETER (MAXDP = 1000,MAXPHS = 5,NW = 10000000,MAXCS=100000)

      INTEGER IPCPH2(MAXPHS+1),IPDPH2(MAXPHS+1)
      DOUBLE PRECISION CSTAT2(MAXCS),DPARM2(MAXDP),W2(NW)
      INTEGER NUMZV,LNZVEC,NUMPV,LNPVEC
      PARAMETER(LNZVEC = 50,LNPVEC = 10)
      DOUBLE PRECISION ZVEC(LNZVEC),PVEC(LNPVEC)
      DOUBLE PRECISION T,TZERO,TFINAL
      COMMON CSTAT2,IPCPH2,DPARM2,IPDPH2,W2

      IPHASE = 1

      CALL OCSEVL(MAXPHS,CSTAT2,MAXCS,IPCPH2,DPARM2,MAXDP,IPDPH2,
     +     W2,NW,IPHASE,T,TZERO,TFINAL,ZVEC,NUMZV,LNZVEC,
     +     PVEC,NUMPV,LNPVEC,IER)

      VHAT = ZVEC(15)

      RETURN
      END


C Function to extract MEDS solution YBAR for MI phase

      DOUBLE PRECISION FUNCTION YBAR(T)

      INTEGER MAXPHS,NW,MAXCS,MAXDP,IPHASE,IER
      PARAMETER (MAXDP = 1000,MAXPHS = 5,NW = 10000000,MAXCS=100000)

      INTEGER IPCPH2(MAXPHS+1),IPDPH2(MAXPHS+1)
      DOUBLE PRECISION CSTAT2(MAXCS),DPARM2(MAXDP),W2(NW)
      INTEGER NUMZV,LNZVEC,NUMPV,LNPVEC
      PARAMETER(LNZVEC = 50,LNPVEC = 10)
      DOUBLE PRECISION ZVEC(LNZVEC),PVEC(LNPVEC)
      DOUBLE PRECISION T,TZERO,TFINAL
      DOUBLE PRECISION C1(2),NB1
      COMMON CSTAT2,IPCPH2,DPARM2,IPDPH2,W2

      PARAMETER (NB1=1.D0)
      DATA C1/1.D0,1.D0/

      IPHASE = 1

      CALL OCSEVL(MAXPHS,CSTAT2,MAXCS,IPCPH2,DPARM2,MAXDP,IPDPH2,
     +     W2,NW,IPHASE,T,TZERO,TFINAL,ZVEC,NUMZV,LNZVEC,
     +     PVEC,NUMPV,LNPVEC,IER)

      YBAR = C1(1)*ZVEC(3) + C1(2)*ZVEC(4) + NB1*ZVEC(12)

      RETURN
      END
```

A.4 Analysis and Presentation of Results

The above SOCS driver creates several data files as outputs. These files are suitable for processing in MATLAB. The following MATLAB m-files process the data files outputted by SOCS to facilitate analysis and visualization of results.

A.4.1 Detection Signal Phase Processing

This m-file reads and plots the minimum energy detection signal and the separability index.

```
%portion of ex1_1to4.m
%ex1_2_*.dat
fid = fopen('ex1_2_1.dat','r');
[A21,cnt] = fscanf(fid,'%22g',[11,208]);
fclose(fid);

fid = fopen('ex1_2_10.dat','r');
[A210,cnt] = fscanf(fid,'%22g',[11,394]);
fclose(fid);

fid = fopen('ex1_2_20.dat','r');
[A220,cnt] = fscanf(fid,'%22g',[11,642]);
fclose(fid);

fid = fopen('ex1_2_100.dat','r');
[A2100,cnt] = fscanf(fid,'%22g',[11,800]);
fclose(fid);

%gamma.dat
fid = fopen('gamma.dat','r');
[G,cnt] = fscanf(fid,'%22g',[3,20]);
fclose(fid);

%extract vhat
v1=A21(11,:);v10=A210(11,:);v20=A220(11,:);v100=A2100(11,:);

%need to rescale the vhats
%divide by the sqrt of the objective function
```

```
%T=100 case gave negative vhat, so resign it
v1=v1/sqrt(A21(7,104));
v10=v10/sqrt(A210(7,197));
v20=v20/sqrt(A220(7,321));
v100=-v100/sqrt(A2100(7,400));

%extract time vector
t1=A21(1,:); t10=A210(1,:); t20=A220(1,:); t100=A2100(1,:);

%plot vhats
plot(t1,v1)
plot(t10,v10)
plot(t20,v20)
plot(t100,v100)
%thesis plots ex121v.eps, ex1210v.eps, ex1220v.eps, ex12100v.eps

%combined plot of [0,20], [0,100] cases
plot(t20,v20,t100,v100)
%thesis plot ex12v20v100.eps

%plot gamma
T=G(1,:);gamma=sqrt(G(2,:));
plot(T,gamma)
%thesis plot ex1gammultT.eps

%rescale the vhats and the time vectors for combined plot of all cases
v1=v1/max(v1);
v10=v10/max(v10);t10=t10/max(t10);
v20=v20/max(v20);t20=t20/max(t20);
v100=v100/max(v100);t100=t100/max(t100);
plot(t1,v1,t10,v10,t20,v20,t100,v100)
%thesis plot ex12multv.eps

%end of routine
```

A.4.2 Model Identification Phase Processing

This m-file reads and plots the normal to the separating hyperplane and ybar.

```
%adex7_2m.m
fid = fopen('adex7_2_1m7.dat','r');
```

```
[Aa71m7,cnt] = fscanf(fid,'%22g',[15,250]);
fclose(fid);

%extract y0 and y1
y17=Aa71m7(15,:); y07=Aa71m7(14,:);

%compute yb7 as the midpoint of segment {(yb1+yb0)/2}
yb7= (y17+ y07)/2;

%extract time vector
te= Aa71m7(1,:);

%construct normal to hyperplane
a7= (y17- y07)/ L2norm(y17-y07,te);

%plot ybar
plot(te,yb7)
%thesis plot ae721yb7.eps

%plot a(t)
plot(te,a7)
%thesis plot ae721a7.eps

%L2norm.m function to compute L2 norm
function x = L2norm(f,t)
%find the L2 norm of a discretized function on [0,tf]
%uses right approximation (as opposed to left/center)
%input vector function f
%input time vector t corresponding to values of f
N = length(f);
if N~=length(t)
   error('vectors must be same length')
end
tr=t(2:N);
tl=t(1:N-1);
dt=tr-tl;
fsq=f(1:N-1).^2;
x=sqrt(fsq*dt');

%end of routine
```