Advanced Sliding Mode Controllers
For Industrial Applications

A. I. Bhatti

Abstract

This thesis deals with the industrial application of sliding mode controllers. Sliding mode controllers based on both linear models and nonlinear models are considered. Special attention is paid to the nonlinear modelling of the systems for sliding mode controller design. The possibility of using neural networks for model generation is explored. Novel schemes for uncertainty bounds estimation are introduced and subsequently used for robust sliding mode controller design. Later, a novel approach for sliding mode based parameter estimation for a nonlinear model with known structure but unknown parameters is introduced. This parameter estimation scheme is integrated with sliding mode controller design to provide an overall controller design framework. The stability of these schemes is proven through quadratic stability concepts.

The sliding mode controller design frameworks mentioned above are verified and tested on challenging industrial examples. The temperature control of a high temperature multiburner industrial furnace is a highly coupled and extremely nonlinear problem. A multiburner furnace nonlinear simulation facility is established and used for linear identification and subsequently linear model based sliding mode controller testing. For comparison purposes a two degree of freedom $H_\infty$ controller is also designed and tested. Then a nonlinear model based controller is tested on a single burner furnace simulation. Idle speed control of an automobile engine is an extremely difficult control problem characterised with severe nonlinearities, gross disturbances and huge time delays. A sliding mode controller is designed for this problem and successfully implemented on a test rig. Later on, a nonlinear model based sliding mode controller is designed for the same problem and successfully tested.
ADVANCED SLIDING MODE CONTROLLERS

FOR

INDUSTRIAL APPLICATIONS

Thesis submitted for the degree of
Doctor of Philosophy
at the University of Leicester

by

Aamer Iqbal Bhatti B.Sc. (Lahore), M.Sc. (London)
Department of Engineering
University of Leicester

July 1997
To my parents
Acknowledgements

Certainly its a pleasant obligation to thank my supervisor Dr. S. K. Spurgeon for her encouragement, support and guidance during the course of this research. Working under her supervision has been surely a nice experience. I would further like to thank Dr. D. W. Gu for many useful discussions. I am particularly grateful to Dr. C. Edwards who introduced me to the field of sliding mode control with all its intricacies and nice aspects. Special thanks go to Dr. X. Y. Lu for many useful and fruitful discussions. I would like to thank all the members of the Control Group who have made working in the group a memorable and pleasant experience especially Ghassan, Alex, Song Ju, Li Quin, Abdul, Tim, Jens, Raza, Bag, Ahmet, Fu Wah, Mark, Mathew and Randall. I would specially like to thank Shehzad Jehangir for all the nice time we spent together. I am short of words in acknowledging the love and support given to me by my father, who did not live long enough to see this thesis get finished, though he would be the happiest person if he were around. I would like to thank my mother, brothers for their support and encouragement. My wife also deserves special thanks as she endured with patience my writing up period. At the end I express my gratitude to the Government of Pakistan for funding this research. I am also grateful to Dr. Robert Dorey for his help during the Ford project.

Leicester
July 1997
Papers Pertaining to the Thesis

Published Papers.


Unpublished Papers.


## Contents

1 Introduction .................................................. 1
   1.1 Overview ................................................. 1
   1.2 Thesis contribution ..................................... 4
   1.3 Thesis structure ........................................ 6

2 Sliding Mode Controller Design ....................... 8
   2.1 Introduction ............................................. 8
   2.2 Basic concepts of sliding mode theory .......... 10
      2.2.1 Hyperplane design ................................ 10
      2.2.2 Reachability condition ......................... 11
   2.3 Design example ......................................... 12
   2.4 Sliding mode design procedure .................... 16
      2.4.1 Motivation for using linear models .......... 16
      2.4.2 Observer selection ............................... 16
      2.4.3 Controller Formulation ......................... 18
      2.4.4 Nonlinear Observer Formulation .............. 22
   2.5 Controller design issues ............................ 24
   2.6 Summary ................................................ 25
3 Sliding Mode Control in Nonlinear Systems

3.1 Introduction ........................................................................................................... 26
3.2 Indirect Sliding Mode Controller Design ............................................................ 28
   3.2.1 Model representation .................................................................................. 28
   3.2.2 Sliding Surface .......................................................................................... 29
   3.2.3 Reachability condition .............................................................................. 30
   3.2.4 Design Method ........................................................................................... 31
   3.2.5 Advantages of the Indirect Approach to Sliding Mode Control ............. 32
3.3 Design Example: Underwater Vehicle Position Control ..................................... 33
   3.3.1 Controller design ..................................................................................... 33
   3.3.2 Closed loop simulation ............................................................................ 34
3.4 Direct Sliding Mode Control ................................................................................ 37
   3.4.1 Introduction .............................................................................................. 37
   3.4.2 System model with uncertainty ................................................................ 38
   3.4.3 Direct Sliding Surface ............................................................................... 39
   3.4.4 Design Method .......................................................................................... 40
   3.4.5 Advantages of the direct approach to sliding mode control ................. 42
3.5 Design Example: Direct sliding mode design for underwater vehicle ............... 42
   3.5.1 Calculation of the uncertainty bounds ...................................................... 42
   3.5.2 Controller design ..................................................................................... 44
   3.5.3 Simulation results .................................................................................... 44
3.6 Comparison .......................................................................................................... 48
3.7 Summary ............................................................................................................... 49

4 Sliding Mode Control Based On Neural Networks 51
4.1 Introduction ........................................................................................................... 52
4.2 Literature review ............................................................................................... 52
  4.2.1 Nonlinear identification using neural networks ........................................... 53
  4.2.2 Use of neural networks in variable structure control .................................... 54
4.3 Formulation of feedforward neural networks ............................................. 55
4.4 Backpropagation training algorithm with robust cost function ................. 57
4.5 Combination of the sliding mode controller and neural network model .... 60
  4.5.1 Initial condition issues .............................................................................. 60
  4.5.2 Structural issues ...................................................................................... 61
4.6 Design Example: Output tracking for a Tunnel Diode circuit using indirect sliding mode with neural model ....................................................... 62
4.7 Determination of robustness bounds for neural network based direct sliding mode controller ................................................................. 69
  4.7.1 Uncertainty bound estimation using curve fitting ...................................... 69
  4.7.2 Uncertainty bound estimation from Huber's function ............................ 71
  4.7.3 Uncertainty bound estimation using network distance .......................... 75
  4.7.4 Statistical estimation of the uncertainty bound ........................................ 76
4.8 Summary ........................................................................................................... 78

5 Sliding Mode Based Parameter Estimation ...................................................... 80
5.1 Introduction ....................................................................................................... 80
5.2 Literature review ............................................................................................. 80
5.3 Sliding mode based robust parameter estimation .......................................... 82
5.4 Design example ............................................................................................... 87
5.5 Sliding mode based parameter estimation and control ............................... 88
5.6 Design example ............................................................................................... 91
6 High Temperature Multi-burner Furnace Control

6.1 Introduction ................................................................. 95
6.2 Multi-burner Furnace Control: Literature Review ................... 96
6.3 Multi-burner Furnace Simulation Code Development ............... 97
  6.3.1 Furnace Simulation Algorithm .................................. 97
  6.3.2 Flow Calculation ..................................................... 98
  6.3.3 Load Movement ...................................................... 98
6.4 Controller Implementation ................................................ 99
6.5 Linear Identification ...................................................... 99
  6.5.1 Furnace Configuration Selection ................................. 99
  6.5.2 Operating Point Selection ......................................... 101
  6.5.3 Preliminary Identification Experiments ....................... 101
6.6 Model Estimation .......................................................... 107
6.7 Analysis of the linear model .............................................. 109
6.8 Linear model based sliding mode controller design .................. 111
  6.8.1 Furnace Controller Design Issues ............................. 111
6.9 Nonlinear Closed loop Simulation ...................................... 113
6.10 $H_\infty$ Controller Design ........................................... 115
  6.10.1 Two Degree of Freedom Design ............................... 116
  6.10.2 Loop Shaping Design Procedure ............................. 117
  6.10.3 Controller Design ................................................ 118
6.11 Comparative Analysis .................................................. 122
6.12 Direct sliding mode controller design ................................ 123
Chapter 1

Introduction

This Chapter introduces the structure and contribution of the thesis. The thesis covers a wide range of issues pertaining to the industrial application of the sliding mode controllers such as sliding mode controller design, neural network modelling, adaptive parameter estimation, high temperature multiburner furnace control and automotive engine speed control. Owing to the diverse nature of these topics, the literature survey for each topic is subsequently presented in the corresponding Chapter.

Section 1.1 motivates the need for applying sliding mode control to solve industrial problems, along with a general statement of the thesis contribution. The individual contributions are highlighted individually in Section 1.2, followed by the description of the layout and the structure of the thesis in Section 1.3.

1.1 Overview

For various reasons, today's industry is undergoing a process of fundamental transformation. Pollution control to meet strict regulations (for instance, emission control in automobiles), tougher quality control objectives to face cut throat market competition, production efficiency to decrease the overall cost, are some of the factors contributing to this change. These trends have increased the need for the application of sophisticated control techniques in industry. The increased use of digital instrumentation and control has rendered the implementation of advanced control methods an economically viable proposition. On the contrary, the level of acceptance of advanced controller de-
sign techniques is surprisingly low. Even eminent engineers do not attempt to hide their lack of enthusiasm about new control techniques. It would be unfair to dismiss their reservations totally on the grounds of ignorance and conservatism. It is commonly believed in the industrial community that advanced controllers are very complex and hence are abstract and impracticable. These controllers do not provide the extent of transparency needed for day to day operations. In the first place, many industrialists are of the opinion that these controllers are hard to design, with a complex structure that is difficult to implement. Furthermore, the tuning process is thought to be not very straightforward. Another hurdle is the popularity and wide acceptability of PID controllers. It would not be wrong to say that the only good thing about PID is that it works. That said, the practical limitations of PID controllers are not very difficult to find. PID controllers can provide robustness, however, accompanied with the degradation in the performance. Usually, for a control application, a set of PID controllers is tuned with every PID tuned to a different process condition. Banks of PID configurations with individually tuned PID controllers for every part of the operation envelope are not very hard to find in industry. In coupled MIMO processes decoupling cannot be easily achieved through PID tuning. Furthermore, PID does not provide an optimal (or even suboptimal) solution to a control problem. In optimisation terms, PID controllers are a sort of false minimum in which the modern day industry is trapped.

On the other hand, there is a lot of work needed to make sophisticated controllers acceptable to industry. Though considerable work has been done towards industrial application of new control concepts (as apparent from the references given in the later chapters), there are various points lacking in the literature. Usually, in a control application, the emphasis is on the verification of the design technique itself. The design details like specification translation, use of extra degrees of freedom and different parameter selections are often not elaborated. This makes it unappealing for an industrial engineer, as s/he would not be able to correlate the design concepts to the real world. Sometimes the controller representation is made unnecessarily complicated by the lack of distinction between the analysis and implementation issues. Also many assumptions adopted during controller design are simply not practical. Seemingly, it is not far from the truth to state that there is a wide gap between the industry and control community.
Chapter 1. Introduction

The main topic of this thesis is to devise new controller design frameworks which take into account all the phases of a controller design process, beginning from system modelling to controller implementation and tuning in an actual industrial environment. The sliding mode controller design technique is specifically chosen for this purpose for various reasons. Firstly, it is a nonlinear approach which has the capability to accommodate nonlinear systems. The technique is inherently robust against a wide class of model uncertainties. The design principles involved are relatively straightforward as compared to other design methods. It has been attempted in the thesis to cover all the aspects of control design and implementation for an industrial process. A novel framework for nonlinear controller design is proposed. The first step in a controller design process is to establish models of the system. In general, it is very difficult to find existing models of the industrial process being considered. Even if such a model exists, most often it is unsuitable for controller design purposes. Linear model identification methods are well established now. The practical issues which can be encountered in the industrial environment are considered. Two new methods for nonlinear model determination are proposed and demonstrated through design examples. There are two well-defined traits in the sliding mode literature. The first category of sliding mode controllers uses linear models with uncertainties for controller synthesis while the other is based on nonlinear models. Two design schemes recently proposed by Edwards and Spurgeon [20] and Lu and Spurgeon [49] are discussed and the issues regarding their industrial applications are considered. Furthermore, transparent design procedures of these schemes are devised. These design procedures are finally applied to a number of industrial examples. Industrial processes can be broadly categorised into infinite dimensional systems and finite dimensional ones. To demonstrate the utility of sliding mode controllers in industry it is necessary that the controller should exhibit the properties of robustness, disturbance rejection, decoupling and output tracking. Keeping these things in mind, two specific industrial processes were chosen. Multiburner high temperature furnaces represent infinite dimensional systems with very strong coupling. The first application is the temperature control of a multiburner furnace for a demanded temperature profile varying in time and space. The second application is concerned with idle speed control of a car engine, which is a finite dimensional system with heavy disturbances. The
thesis shows that sliding mode controllers are not high gain and chattering is not a problem, as is often thought. An $H_\infty$ controller is also designed and implemented for the furnace and it is shown by comparison that a sliding mode controller is not using a particularly high control effort. A new strategy for the rejection of known disturbances is also proposed. The contributions of the thesis are outlined in the next section.

1.2 Thesis contribution

The main contribution of the thesis is that it has advanced the industrial applicability of sliding mode controllers. The following individual contributions lead to this objective:

1. In the field of nonlinear model based sliding mode controller design:

   (a) A novel way to identify nonlinear continuous time models using neural networks is proposed.

   (b) Backpropagation theorem is the most frequently used training algorithm for neural network training. The cost function used in the backpropagation algorithm is a quadratic function of error. Huber [62] proposed a cost function for the least square estimation techniques called Huber's function. He claimed that the function is robust against the non-Gaussian noise. For the first time, the backpropagation algorithm is derived with Huber's function as a cost function.

   (c) Robust sliding mode controller design scheme proposed by Lu and Spurgeon [49] is applied to the proposed neural network based continuous time nonlinear model.

   (d) For robust sliding mode controller design, estimates of uncertainty bounds are needed. Various ways to determine these bounds are proposed.

   (e) A new sliding mode based robust parameter estimation scheme for nonlinear model determination is proposed with stability and convergence analysis.

   (f) A novel adaptive robust sliding mode controller based on the above mentioned model is proposed with stability and convergence analysis.
Chapter 1. Introduction

2. Linear model based sliding mode controller design scheme of Edwards and Spurgeon [20] is extended, so that:

   (a) Direct hyperplane design using LQ and minimum entropy design techniques is derived.

   (b) Nonlinear parameter selection and tuning is correlated to the controller performance.

3. In the field of industrial application these problems are solved:

   (a) Multiburner furnace simulation code modification and development.

   (b) Simulation code is used to decide a furnace configuration suitable for control purposes.

   (c) Criteria are established for linear identification of a multiburner furnace.

   (d) A linear model based sliding mode controller is designed for furnace temperature control.

   (e) Furnace temperature control is considered for a given temperature profile varying in time and space.

   (f) Disturbance rejection in furnace temperature control is established.

   (g) A nonlinear model based furnace controller is designed.

   (h) A continuous time $\Phi$ based automotive model is developed.

   (i) A linear model based sliding mode controller is designed for idle speed control of the automobile engine.

   (j) Sliding mode controller implementation in dSPACE is considered.

   (k) Nonlinear simulation and successful rig trials are used to test the controller

   (l) A nonlinear model based sliding mode controller is designed for the car engine.

The next section overviews the whole thesis chapter by chapter.
1.3 Thesis structure

Chapter 2 is a preliminary Chapter introducing the basic concepts of the sliding mode controller design method. A complete literature survey of the fundamentals of sliding mode design is presented and illustrated through a design example. A linear model based sliding mode controller-observer design method proposed in [20] is also introduced and the issues regarding industrial application are discussed.

Chapter 3 is the second of the two preliminary Chapters of this thesis. In Chapter 3 nonlinear model based sliding mode design techniques are introduced. Two particular design methods proposed in [49] are discussed in detail. The complete design procedures are described and demonstrated through design examples.

In Chapter 4 a novel way of using neural networks for nonlinear model determination is proposed so that the controller design techniques proposed in Chapter 3 can be used. Then the sliding mode controllers are designed using the neural network based model. The robust sliding mode controller needs uncertainty bounds for its design. Various novel ways to achieve this are proposed and demonstrated through design examples. The choice of uncertainty bounds to reduce chattering in the controller is investigated.

Chapter 5 proposes a new method of parameter estimation for a nonlinear model with unknown parameters but known structure. This is illustrated through a design example. Next, the controller and parameter estimation design scheme is integrated. The stability and convergence proofs for both schemes are given.

High temperature furnace control is considered in Chapter 6. After details of the nonlinear furnace simulation code are given, linear identification of the furnace is performed. Various practical issues relating to the furnace identification are discussed. A linear model based sliding mode controller is designed and tested on the nonlinear model. For comparison purposes, an $H_\infty$ controller is also designed and tested. A quantitative comparison is made later on. Finally, a controller is designed and tested using the techniques introduced in Chapter 5.

Idle speed control of an automobile engine is addressed in Chapter 7. After giving a detailed design procedure for linear model based sliding mode controller design based on the theoretical framework of Edwards and Spurgeon [20], including hyperplane design, observer design, nonlinear parameter selection, the controller is implemented in
the implementation environment dSPACE. Later on, the problem is considered in a SISO framework. The nonlinear simulation and rig trials are presented and analysed. Finally, a controller is designed and tested using the techniques from Chapter 5. Finally, the thesis is summarised and concluded in Chapter 8. Future work emanating from the course of this research is also presented.
Chapter 2

Sliding Mode Controller Design

This chapter is aimed at introducing the basic concepts of the sliding mode control theory. This is essentially a preliminary Chapter. First a thorough literature survey of the linear approaches to sliding mode control theory is presented. Then the fundamentals of the sliding mode theory are explained and illustrated through a design example. The introduction of these concepts provides the necessary background to explain the sliding mode controller-observer design formulation proposed by Edwards and Spurgeon [20]. The practical issues regarding this scheme will be highlighted. This Chapter sets the stage for the sliding mode controller design applications covered in Chapters 6 and 7.

2.1 Introduction

The control design methods commonly referred to as sliding mode techniques appear in the control literature since the late fifties. From the outset, the method was highlighted as a robust control technique, when most of the emphasis was on optimal controller design (modern control). In addition to robustness, it provides another advantage above other robust controllers; the technique is entirely nonlinear in nature. Consequently, the span of this design method does not limit itself to the domain of systems which are easily expressible through linear models, rather it offers the promise to provide generic controller design frameworks applicable to wide classes of nonlinear systems. The basic idea behind sliding mode control is to specify an observable function of the system states which may be regarded as a fictitious output, and then to design a controller to regulate
Chapter 2. Sliding Mode Controller Design

this dummy output. Following regulation of the function to zero, the system will behave in accordance with the function parameters selected. If the function selected is linear, then after the regulation of this function, the nonlinear system will exhibit linear dynamics determined by the selected function. Sometimes, this function can also consist of the output errors and their integrals. Hence by regulating the specified function it is possible to regulate the errors.

Emylianov and his colleagues proposed and highlighted sliding mode controller design for the first time in the late fifties [33]. After this, the theory was extended and developed in various dimensions. The popularity of the technique gained momentum because of its natural appeal for application to a wide class of systems containing discontinuous elements as their control inputs like relays or Pulse Width Modulation (PWM) switching. A generic sliding mode controller for linear systems with bounded uncertainty was proposed by Ryan and Corless [68]. This proposition was developed by Spurgeon and Davies [13] to incorporate output tracking into the controller design. Burton and Zinober [6] introduced a smoothing factor into the control scheme. Zinober et al [6, 15, 101, 110] also devised methods to incorporate desired dynamic response into the sliding mode framework. This controller eventually needs full state information for its design and implementation which motivates the need for an observer. Utkin, and later on Breinl and Leitman presented an asymptotic observer with a sliding mode controller [5]. Walcott and Zak [94] combined the controller and the observer together and gave intuitive arguments that the closed loop will be stable. Edwards and Spurgeon [20] performed a rigorous closed loop analysis and set the conditions for which the closed loop would be stable. The literature overview of the sliding mode controller designed with nonlinear models will be presented in Chapter 3.

Until now, this controller-observer pair has been mainly investigated for its stability. The relevant theory has now passed the stage of theoretical assessment. The theory needs to be tested against the harsh environment of industrial processes. The scheme was implemented on a single burner furnace [19] in a SISO framework. It is logical that the controller should be implemented and assessed for a MIMO industrial system. Furthermore, the determination of the linear and nonlinear parameters of the controller-observer pair needs further investigation i.e how to explore and utilise free-
doms in controller design while keeping in view the given specifications. It will be attempted to present a real design formulation involving this particular scheme which will be useful to the users in industry. This design framework should be transparent for the end user so as to pave the way for the full utilisation of sliding mode control in industry. The degrees of freedom involved in the design will be related to commonly known controller design issues like system bandwidth, steady state performance, robustness, and disturbance rejection. An industrial engineer needs to specify such characteristics through controller parameter selection, and by using the proposed design framework he should be able to get a sliding mode controller in readily implementable form.

After introducing the basic concepts of sliding mode theory in Section 2.2, it is illustrated through a design example in Section 2.3. A complete procedure for sliding mode controller design is introduced in Section 2.4. Practical issues regarding controller design are attended in Section 2.5. Finally, the chapter is summarised and concluded in Section 2.6.

2.2 Basic concepts of sliding mode theory

To simplify the description consider a nominal linear state space model:

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Bu(t) \\
y(t) &= Cx(t)
\end{align*}
\]  

(2.1)

where \( x \in \mathbb{R}^n \) is the system state vector, \( u \in \mathbb{R}^m \) is the input vector, \( y \in \mathbb{R}^p \) is the output vector. A variable \( s \in \mathbb{R}^m \) is chosen as a function of the system states. The most popular choice is to choose a linear combination of system states:

\[ s = Sx \]

where \( S \in \mathbb{R}^{m \times n} \) defines a manifold (so called sliding surface) in the system state space. The controller is designed in two phases:

2.2.1 Hyperplane design

The system is said to be in sliding motion when:

\[ Sx = 0 \]
Proper choice of sliding manifold, characterised by $S$, will determine the system dynamics in the sliding mode. Since the dynamics pertaining to this phase lie in the null space of $S$, they are termed as null space dynamics. During sliding, the motion of the system in its state space is restricted to the hyperplane defined by $Sz = 0$, so the system exhibits the behaviour of a linear system characterised by the choice of $S$. This design is called the hyperplane design. During the sliding motion the system should exhibit desired dynamics. This is achieved by the proper design of the sliding surface i.e. through hyperplane design. For linear systems, the hyperplane design determines the eigenvalues of the system during sliding motion. This step will be further explained through an example.

### 2.2.2 Reachability condition

Once the system is in sliding motion, the controller should be designed in such a way so that it is able to keep the system in sliding motion despite the presence of uncertainties. The prime task of the controller is to regulate the sliding variable i.e. the controller should drive the system into sliding motion. This is achieved by setting the condition on the control law under which the system will reach the sliding manifold. Such a condition is called the reachability condition. When the system is not sliding the sliding function $s$ will lie in the range space of $S$. Hence the dynamics related to the phase when the $s \neq 0$ (reachability phase) are called range space dynamics. The reachability condition is chosen to make the sliding manifold attractive to the remaining state space so that if the system is not on the sliding surface it should move towards the surface. The usual approach is to perform quadratic stability analysis in presence of uncertainties. For the regulation of the sliding variable $s$ a candidate Lyapunov function can be proposed as:

$$V = \frac{1}{2} s^T s$$

implying that:

$$\dot{V} = s^T s$$

Hence, $s$ will decay to zero only if:

$$s^T s < 0 \quad (2.2)$$
Therefore, the reachability condition consists of a relation between the sliding variable $s$ and its higher derivatives so that the above mentioned condition is satisfied. For a scalar $s$ the most commonly used one is:

$$\dot{s} = -k_ds\text{sign}(s) \quad (2.3)$$

This reachability condition will make sure that:

$$s\dot{s} < 0$$

hence making the sliding manifold attractive. This kind of discontinuous reachability condition will introduce chattering in the controller action. To avoid chattering a linear reachability condition can also be defined, such as:

$$\dot{s} = -k cs \quad (2.4)$$

This condition will asymptotically regulate $s$. The rate of decay will be controlled by the choice of $kc$.

### 2.3 Design example

Consider a simple double integrator system:

$$\dot{x}_1 = x_2 \quad (2.5)$$

$$\dot{x}_2 = u + f(x,t) \quad (2.6)$$

Here $u$ is the input and $f(x,t)$ is the uncertainty acting through the input channel (usually called matched uncertainty). All the variables considered are scalars. $f(x,t)$ is not known but bounded such that:

$$|f(x,t)| < r$$

where $r$ is a positive scalar. The sliding surface is defined as:

$$s = cx_1 + x_2$$

This step is essentially the hyperplane design. The choice of $c$ determines the speed of dynamics during sliding motion. The reachability condition is specified as:

$$\dot{s} = -k_ds\text{sign}(s) - kcs \quad (2.7)$$
where $k_d$ and $k_c$ are strictly positive scalars. In the absence of uncertainty $f(x, t)$ it is obvious that the selected reachability condition satisfies the inequality (2.2):

$$\dot{s} = cx_2 + \dot{x}_2$$

hence:

$$s\dot{s} = -k_d|s| - k_c s^2 < 0$$

by utilising the property:

$$|s| = \text{sign}(s)s$$

The control $u$ can be computed from 2.7 as:

$$u = -cx_2 - k_d \text{sign}(s) - k_c s$$

In the presence of $f(x, t)$, the perturbed $\dot{s}$ is:

$$\dot{s}_{\text{per}} = \dot{s} + f(x, t)$$

hence:

$$s\dot{s}_{\text{per}} = s[f(x, t) - k_d \text{sign}(s) - k_c s]$$

$$= sf(x, t) - |s|k_d - s^2 k_c$$

$$< |s|r - |s|k_d - |s|^2 k_c$$

$$= -|s|[-r + k_d + k_c |s|]$$

It can be deduced that the inequality (2.2) is true for $\dot{s}_{\text{per}}$ only if:

$$k_d + k_c |s| > r$$

or a relaxed condition could be:

$$k_d > r$$

The sliding is guaranteed if $k_d > r$. It can be inferred that higher gain $k_d$ will increase robustness for a given $k_c$. Increasing $k_d$ unnecessarily can lead to undesirable chattering. Figure 2.1 shows the phase plane when there is no uncertainty present or in other words $f(x, t) = 0$. The parameters used are:

$$k_c = 0;$$
Figure 2.1: Phase plane with no uncertainty present

\[ k_d = 1 \]
\[ c = 1; \]

Figure 2.1 clearly shows the two phases of system behaviour. In the first phase the controller drives the system onto the sliding surface (shown by dotted lines). The system hits the surface at point A. After reaching the sliding surface the system stays there. When uncertainty is introduced in the form of \( f(x, t) = 0.9\sin(\omega t) \), the chattering is evident, as shown in the Figure 2.2. In this case the upper bound on the uncertainty \( r \) is 0.9 which is less than \( k_d \) chosen. If the uncertainty is chosen to be a constant greater than \( k_d \), the system cannot reach the sliding surface as shown in the Figure 2.3. \( r \) is selected to be 2 in this case.

Having explained the basic concepts of sliding mode control theory, a generic scheme for designing a sliding mode controller based on linear models will be introduced.
Chapter 2. Sliding Mode Controller Design

Figure 2.2: Phase plane with uncertainty $f(x, t) = 0.9\sin(\omega t)$

Figure 2.3: System is not reaching the sliding mode when $r > k_d$
2.4 Sliding mode design procedure

2.4.1 Motivation for using linear models

It is quite common that a given nonlinear system is approximated by a linear model. There are several reasons for adopting a linear model even though it is known that the system is essentially nonlinear. In industry, it is often not possible to determine a nonlinear model of the plant to be controlled. There could be two possible ways to determine a nonlinear model for a given plant. The first one is to construct a model from physical principles involved in the industrial process (usually called first principle basis). This is an extremely expensive process in terms of time and money. Even, at the end of the day, if a nonlinear model is obtained it would be so large and full of redundancies that it cannot be readily used for controller design. Hence, this way is often impractical for controller design. The second way could be to use nonlinear identification techniques such as neural networks or bilinear identification to get such a model. The snag with this approach lies in the lack of generality and user oriented identification methods. It is next to impossible to convince industry to acquire and use nonlinear models for controller design. On the contrary, a linear model can be obtained using well established linear identification techniques, hence providing easily obtainable low cost models for controller design. These identification techniques will be discussed and used when designing a controller for a high temperature multiburner furnace in Chapter 6. In addition, for mechanical finite dimensional systems, the given nonlinear models can be linearized to utilise well established linear control system analysis and design tools. Therefore it is logical to describe a sliding mode controller design technique based on linear models to be used for the control of industrial processes.

2.4.2 Observer selection

Availability of full state information is assumed in the design of the linear model based sliding mode controllers mentioned in Section 2.2. Implementation of such a controller requires that all the states of the linear model are measurable, which is seldom the case. Usually, only a few states are measurable. In the case of the models obtained through linear identification the states do not have any physical meaning and are hence unmea-
surable. In either case, it is necessary to estimate the system states from the measurable signals of the plant (outputs). For state estimation, observers are used. A Luenberger or asymptotic observer [23] can be used in conjunction with the sliding mode controller. But a sliding mode observer would be a better choice, as it would be robust against a certain class of uncertainties [18]. It can be argued that a Kalman filter can also be utilised as a robust observer. But a Kalman filter assumes statistical knowledge of the uncertainties, which could be available in aerospace industries but certainly not in process industries [28].

It is decided to use the sliding mode controller-observer pair proposed by Edwards and Spurgeon [20], as they have rigorously shown that the closed system will be stable. Furthermore, this scheme has already been tried on industrial plants [19]. In the remaining part of this section the problem of designing a robust nonlinear sliding mode controller based on a linear model will be considered. This technique is robust against a certain class of uncertainties. For this controller design, full state information is needed, so a nonlinear sliding mode observer design will also be discussed.

A nonlinear system can be approximated as:

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Bu(t) + f(t, x, u) \\
y(t) &= Cx(t) \\
f(t, x, u) &= B(g_1(t, x, u)u + g_2(t, x))
\end{align*}
\] (2.8)

where \( x \in \mathbb{R}^n \) is the system state vector, \( u \in \mathbb{R}^m \) is the input vector, \( y \in \mathbb{R}^p \) is the output vector. The function \( f(t, x, u) \) represents the uncertainties in a nominal linear model. Note that \( f(t, x, u) \) is assumed unknown but bounded i.e \( \|g_1(t, x, u)\| \leq k_{g1} \) and \( \|g_2(t, x)\| \leq \alpha(t, y) \) for some known function \( \alpha(t, y) \) and known scalar \( k_{g1} \). Variable Structure Control with a sliding mode is well known for its robustness properties. These will be exploited using the nonlinear controller-observer scheme formulated in [19]. As the output is desired to track certain reference signals, a tracking arrangement is incorporated. For the sake of generality, the demanded outputs \( r(t) \in \mathbb{R}^p \) are taken as time dependent. It is assumed that:

1. The plant is square \( (m = p) \)
2. \( \det(CB) \neq 0 \)
3. Any invariant zeros of the nominal linear system are in the left half of the complex plane.

If Rosenbrock's system matrix is denoted by:

\[ R_b(q) = \begin{bmatrix} qI - A & B \\ -C & 0 \end{bmatrix} \]

then invariant zeros for the linear system characterized by the triplet \((A, B, C)\) are defined as:

\[ q \in \mathbb{C} : \det R_b(q) = 0 \]

2.4.3 Controller Formulation

For controller design a linear model of the system is needed. A transformation \( z = T x \) is obtained through QR decomposition to express the system in the canonical form:

\[
A_z = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B_z = \begin{bmatrix} 0 \\ B_2 \end{bmatrix}, \quad C_z = \begin{bmatrix} C_1 & C_2 \end{bmatrix}
\]

(2.9)

where \( A_{11} \in \mathbb{R}^{(n-p) \times (n-p)} \), \( B_2 \in \mathbb{R}^{m \times m} \) and \( C_2 \in \mathbb{R}^{p \times p} \). The dynamic demand profile is defined as:

\[
\dot{w}(t) = \Psi(w(t) - W)
\]

(2.10)

where \( \Psi \in \mathbb{R}^{p \times p} \) is a stable design matrix and \( W \) is a constant vector. This equation represents an ideal reference model where \( w(t) \) defines a dynamic profile which ultimately converges to the demand vector \( W \). The significance of \( \Psi \) will be discussed later. If the tracking error \( e \) is:

\[
e(t) = w(t) - y(t)
\]

(2.11)

then the integral of the tracking error, \( \dot{z}_e(t) \), is given as:

\[
\dot{z}_e(t) = w(t) - y(t)
\]

(2.12)
These integral states are introduced to provide a framework for output tracking. An augmented state vector is defined as follows:

\[
z_a = \begin{bmatrix} z_e \\ z \end{bmatrix}
\]  

(2.13)

The augmented state vector has the dimensions \( p+n \). The first \( p \) states are the integrals of output error. The last \( n \) states are the actual system states. Now the augmented state vector is repartitioned to isolate the input channels:

\[
z_a = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}
\]  

(2.14)

where \( z_1 \in \mathbb{R}^n \) and \( z_2 \in \mathbb{R}^m \). Now \( z_1 \) is defined to have the first \( n \) states of \( z_a \) and \( z_2 \) contains the last \( m \) states of \( z_a \). Considering the structure of \( B \) it is obvious that the inputs are affecting the states in \( z_2 \) only while \( z_1 \) contains the integral error states and the first \( n-m \) states of the original system. Since \( p = m \), the first and second partitions are compatible. The newly defined augmented system can be rewritten in the form:

\[
\begin{align*}
\dot{z}_1(t) &= \hat{A}_{11}z_1(t) + \hat{A}_{12}z_2(t) + \hat{T}w(t) \\
\dot{z}_2(t) &= \hat{A}_{21}z_1(t) + \hat{A}_{22}z_2(t) + B_2u(t)
\end{align*}
\]  

(2.15)

where \( \hat{T} \) is the distribution matrix for the demand signal:

\[
\hat{T} = \begin{bmatrix} I_p \\ 0 \end{bmatrix}
\]

The new partitioned state matrix is given as:

\[
\hat{A} = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ \hat{A}_{21} & \hat{A}_{22} \end{bmatrix} = \begin{bmatrix} 0 & -C_1 & -C_2 \\ 0 & A_{11} & A_{12} \\ 0 & A_{21} & A_{22} \end{bmatrix}
\]  

(2.16)

A possible sliding surface can be proposed as:

\[
S = \{(x_a) \in \mathbb{R}^{n+p} : s(z_a) = 0\}
\]

\[
s(z_a) = S_1z_1 + S_2z_2 - S_ww
\]  

(2.17)

where \( S_1 \in \mathbb{R}^{m \times n} \), \( S_2 \in \mathbb{R}^{m \times m} \) and \( S_w \in \mathbb{R}^{p \times p} \). \( S_w \) is a design matrix which will be seen to specify the dynamics of the sliding surface for the integral states. If the ideal sliding motion can be attained, then, from equation (2.17):
Chapter 2. Sliding Mode Controller Design

\[
z_2(t) = -S_2^{-1}S_1z_1(t) + S_2^{-1}S_w(t)
\]  

(2.18)

It is possible to choose any \( S_2 \) for the design procedure, so \( S_2 \) can be selected to be invertible. Let

\[
M = S_2^{-1}S_1
\]

(2.19)

then the dynamics when sliding can be obtained by substituting \( z_2(t) \) from equation (2.18) in (2.15):

\[
\dot{z}_1(t) = (\hat{A}_{11} - \hat{A}_{12}M)z_1(t) + (\hat{A}_{12}S_2^{-1}S_w + \hat{T})w(t)
\]

(2.20)

The matrix \( M \), which defines the surface (2.17), is seen to have the role of a state feedback controller for the \( z_1 \) subsystem. For a convenient solution of the reachability problem, a new transformation is introduced as:

\[
\begin{bmatrix}
z_1 \\
\eta
\end{bmatrix} = T_\eta
\begin{bmatrix}
z_1 \\
z_2
\end{bmatrix}
\]

(2.21)

where \( T_\eta \) is given as:

\[
T_\eta = \begin{bmatrix}
I_n & 0 \\
S_1 & S_2
\end{bmatrix}
\]

This transformation essentially means that:

\[
\eta(t) = S_1z_1(t) + S_2z_2(t)
\]

The points in the state space where \( \eta(t) \) is zero are the points where \( s \) is zero. From equation (2.15) the newly transformed system is obtained as:

\[
\begin{align*}
\dot{z}_1(t) &= \bar{A}_{11}z_1(t) + \bar{A}_{12}\eta(t) + \bar{T}w(t) \\
\dot{\eta}(t) &= S_2\bar{A}_{21}z_1(t) + S_2\bar{A}_{22}S_2^{-1}\eta(t) + \Lambda u(t) + S_1Tw(t)
\end{align*}
\]

(2.22)

where:

\[
\bar{A}_{11} = \hat{A}_{11} - \hat{A}_{12}M
\]
\[
\begin{align*}
\dot{A}_{21} &= M\dot{A}_{11} + \dot{A}_{21} - \dot{A}_{22}M \\
\dot{A}_{22} &= M\dot{A}_{12} + A_{22} \\
\dot{A}_{12} &= \dot{A}_{12} + A_{22}S_2^{-1}
\end{align*}
\]

For defining the reachability condition, a form similar to the one defined in (2.4) is adopted. For the linear part of the controller the continuous reachability condition is defined as:

\[
\dot{s} = \Omega s
\]

where \(\Omega\) is a stable design matrix to be specified by the control engineer. The eigenvalues of \(\Omega\) define the range space dynamics of the controller. In other words, these eigenvalues specify the speed at which the system goes into sliding motion. Also, \(s\) can be rewritten as:

\[
s = \eta - S_w w
\]

Using this reachability condition the linear part of the controller can be derived as:

\[
u_L(z_a, w) = H_{eq} z_a + \Lambda^{-1} \Omega s + H_w w + H_\dot{w} \dot{w}
\]

where

\[
\begin{align*}
H_{eq} &= -\Lambda^{-1} S \dot{A} \\
H_w &= -B_2^{-1} M \dot{\hat{T}} \\
H_\dot{w} &= \Lambda^{-1} S_w
\end{align*}
\]

where

\[
\Lambda = S_2 B_2
\]

Here \(\Omega\) specifies the nominal rate of decay at which the \(s\) goes to zero. The linear control law or reachability condition will drive a nominal linear system into sliding motion but in order to make the controller robust against the matched uncertainties a nonlinear control component will also be added. This component will have the form similar to the one proposed by Ryan and Corless [68]. They proposed a unit vector controller which
Chapter 2. Sliding Mode Controller Design

is a scaled version of the normalised sliding surface, as shown later. Assume that $\Omega$ and a symmetric positive definite matrix $P$ satisfy a modified Lyapunov equation:

$$P^{-1}\Omega^T + \Omega P^{-1} = -Q$$  \hspace{1cm} (2.27)

for some positive definite matrix $Q$, then the complete control law is [17]:

$$u = u_L(z_a, w) + u_N$$  \hspace{1cm} (2.28)

$u_L$ denotes the linear and $u_N$ the nonlinear part of the control:

$$u_N = \begin{cases} 
-\rho_N(u_L, y)\Lambda^{-1}\frac{P_o}{||P_o||+\delta_c} & \text{if } s \neq 0 \\ 0 & \text{otherwise} \end{cases}$$  \hspace{1cm} (2.29)

$\delta_c$ is a small positive constant called the smoothing factor and is used to eliminate the chattering in the otherwise discontinuous control action. It is a positive constant and can be tuned if necessary during implementation. $\rho_N$ is a positive scalar function:

$$\rho_N(u_L, y) = \|\Lambda\|\rho_o(u_L, y) + \gamma_c$$  \hspace{1cm} (2.30)

where $\rho_o(u_L, y)$ is a positive scalar function whose magnitude in this instance is determined by the parameters of the nonlinear observer defined later. $\gamma_c$ is a positive scalar.

$$S = [S_1 \ S_2]$$

This control law requires knowledge of all the system states so an observer is required.

2.4.4 Nonlinear Observer Formulation

In this section, a nonlinear discontinuous observer as proposed in [18] will be described. Here $z$ is the state of the transformed system defined in equation (2.9). It has been shown in [17] that if a system can be expressed in regular form then a transformation exists for which the transformed state vector contains the system output as its component. Let:

$$z_o = T_o z$$

where

$$T_o = \begin{bmatrix} I_{n-p} & 0 \\ C_1 & C_2 \end{bmatrix}$$
The transformed system matrix will be:

\[
\hat{A} = \begin{bmatrix}
\hat{A}_{11} & \hat{A}_{12} \\
\hat{A}_{21} & \hat{A}_{22}
\end{bmatrix}
= \begin{bmatrix}
A_{11} - A_{12}C_2^{-1}C_1 & A_{12}C_2^{-1} \\
A_1\hat{A}_{11} - C_2A_{22}C_2^{-1}C_1 & C_1A_{12}C_2^{-1} + C_2A_{22}C_2^{-1}
\end{bmatrix}
\]  

(2.31)

The transformed system is:

\[
\dot{z}_o = \hat{A}_{11}z_o + \hat{A}_{12}y \\
\dot{y} = \hat{A}_{21}z_o + \hat{A}_{22}y + C_2B_2u + f(t, x, u)
\]

Define a corresponding observer by [19]:

\[
\dot{\hat{z}}_o = \hat{A}_{11}\hat{z}_o + \hat{A}_{12}\hat{y} - \hat{A}_{12}e_y \\
\dot{\hat{y}} = \hat{A}_{21}\hat{z}_o + \hat{A}_{22}\hat{y} + C_2B_2u - (\hat{A}_{22} - A_{22}^o)e_y + C_2B_2\nu_o
\]

(2.32)

where \(e_y = \hat{y} - y\) and \(A_{22}^o\) is a stable design matrix. If \(O\) is defined as:

\[
O = \begin{bmatrix}
A_{12}C_2^{-1} \\
A_{22}C_2^{-1} - C_2^{-1}A_{22}^o
\end{bmatrix}
\]  

(2.33)

then the observer can be written as:

\[
\dot{\hat{z}}(t) = A_z\hat{z}(t) + B_zu(t) - OC_z\hat{e}_o(t) + B_z\nu_o
\]

(2.34)

The closed loop observer matrix is:

\[
A_c = A_z - OC_z
\]

and the gain \(O\) is chosen such that \(A_c\) satisfies the Lyapunov equation:

\[
P_cA_c + A_c^TP_c = -Q_o
\]

(2.35)

for some positive definite matrix \(Q_o\). \(P_c\) is a Lyapunov matrix which additionally satisfies the constraint:

\[
P_cB_z = C_z^TL^T
\]

where \(L \in \mathbb{R}^{m \times m}\) is a non singular matrix. If \(P_c\) is a Lyapunov matrix for \(A_{22}^o\) then \(L\) can be explicitly given as \((P_cC_2B_2)^T\) [17]. The scalar function \(\rho_o\) given in the previous section is defined as:

\[
\rho_o(U_L, y) = \frac{k_{s1}\|u_L\| + k_{s1}\gamma_c\|\Lambda^{-1}\| + \alpha(t, y) + \gamma_o}{1 - k_{s1}\kappa(S_2B_2)}
\]
where $\kappa(.)$ is the spectral condition number, $k_{g1}$ is a scalar constant satisfying $k_{g1}\kappa(\Lambda) < 1$. $\gamma_o$ is a tunable positive scalar constant. The nonlinear part of the observer is:

$$\nu_o = \begin{cases} 
-\rho_o(u_L, y)\frac{LC_z e_o}{||LC_z e_o||} & \text{if } LC_z e_o \neq 0 \\
0 & \text{otherwise}
\end{cases} \tag{2.36}$$

if $\varepsilon_1 = \hat{z}_o - z_o$ then the observer error dynamics can be written as:

$$\begin{align*}
\dot{\varepsilon}_1 &= A_{11}\varepsilon_1 \\
\dot{\varepsilon}_y &= \tilde{A}_{21}\varepsilon_1 + A_{22}\varepsilon_y + B_2\nu_o - f(t, x, u) \tag{2.37}
\end{align*}$$

Using a Lyapunov quadratic stability approach, Edwards and Spurgeon [20] have shown that the observer error dynamics are asymptotically stable and that the observer will exhibit a sliding mode on the surface:

$$S_o = \{ e_o \in \mathbb{R}^n : LC_z e_o = 0 \} \tag{2.38}$$

In this way the observer is formulated. This controller-observer pair will not only make the state estimation error go to zero but also the controller will drive the system to track asymptotically the given reference signal. This section has described how to construct a nonlinear controller-observer pair based on certain design variables. The selection of these design variables is the topic of the next section.

### 2.5 Controller design issues

The design decisions associated with developing the controller can be presented in a number of steps.

1. The dynamics of the demand $w(t)$ (the states of the reference model) are specified by the spectrum of $\Psi$. The eigenvalues of $\Psi$ used in equation (2.10) should not be much faster than the open loop plant spectrum to avoid unrealistically large controller effort. To start with, the demand profile spectrum should be close to the open loop plant characteristics. The effect of demand profile on the controller performance will be investigated on real industrial systems. Their relationship with system limitations like valve saturation and coupling will be explored.
2. The sliding surface dynamics are specified by the design matrix $M$ from equation (2.19). The spectrum of $M$ can be determined by different means. Previous knowledge of the plant can be used to define the required closed loop characteristics. This information can be translated into the eigenvalues of $M$. This translation can be done using methods such as LQ design and Minimum Entropy methods. The detailed treatment of these methods can be found in [4]. Using industrial examples a procedure will be presented to design $M$ through a series of steps, starting from well known system performance criterion.

3. The other scalar constants used in the nonlinear part of the controller and the observer are determined from the error bounds of the identified model. Later on, these constants can be tuned during nonlinear simulation to improve performance. The choice of these nonlinear parameters and their effect on the controller performance will be investigated.

4. After establishing the controller and observer parameters it is needed to form a procedure to form an implementable controller for an industrial system. This will be proposed later.

2.6 Summary

In this chapter the fundamental concepts of sliding mode control are described. After reviewing the literature pertaining to the linear model based sliding mode theory, rationale is given for such design. The suitability of such a controller for industrial processes is also motivated. Later on, the theory of sliding mode controller-observer design is explained. Now the stage is set to establish a unified and implementable approach towards industrial controller design, and in turn, to demonstrate the effectiveness of the controller through real industrial controller design. This objective will be accomplished in Chapters 6 and 7.

A significant part of the sliding mode literature is about controller design with nonlinear models. Attention will be focused on the use of nonlinear models for sliding mode controller design in the next chapter.
Chapter 3

Sliding Mode Control in Nonlinear Systems

The previous Chapter provided the necessary background for the industrial application of linear models based sliding mode control theory. This Chapter looks at the nonlinear approaches to the sliding mode control theory suitable for the industrial applications. Two particular sliding mode control schemes proposed by Lu and Spurgeon [50, 47, 49] are described. These two methods have a potential for devising a generic nonlinear controller design method. After presenting the literature review of the nonlinear approaches to the sliding mode controller design the two schemes are formulated in SISO framework. The practical aspects of the two design methods are discussed and illustrated through design examples. The controller robustness against model uncertainties is especially addressed in these examples. One of these design methods determine the controller gains from the knowledge of the uncertainty bounds. The design example presented shows an analytic way of achieving such bounds. The choice of different controller gains to attain robustness as well as low input chattering is also highlighted. Later on, the merits and demerits of both these schemes are discussed.

3.1 Introduction

In the current control literature most of the robust controller design techniques are based on linear models. This represents a serious limitation, as the controller is based on a linear model, so some part of the performance has to be compromised against robustness. It may be expected that a controller will perform better if it is based
on the actual nonlinear model. However it is very difficult to find generic nonlinear controller design techniques. The literature in this field is not as rich as in the case of linear model based controllers. Unlike the linear model case, the nonlinear model has to be in specific canonical form for sliding mode controller design. In the linear case, the only condition is that the model should be in the regular form i.e. the inputs should be isolated from the states of the reduced order system. This isolation is fairly generic, achievable through linear transformation, as shown in equation (2.9). But it is not the case with the nonlinear systems. In order to devise a generic sliding mode controller synthesis method from a nonlinear model the system needs to be in a specific structure or canonical form which is difficult to achieve as system transformation in nonlinear systems is neither straightforward nor generic. People have considered various canonical forms for controller design [33], such as reduced form, controllability form and normal form. Slotine and Sastry [83] considered sliding mode controller design based on an input output decoupled form. They used a sliding surface defined by a Hurwitz polynomial of the output error and its derivatives. Slotine [82] introduced the concept of a boundary layer in the presence of bounded uncertainties in the model where the system is not driven to the sliding manifold but is confined in a boundary around the sliding manifold [84]. The boundary layer concept leads to a continuous control, thus reducing chattering. To ensure robustness the uncertainty bounds were used to calculate the sliding gains. Fu and Liao [24] considered sliding mode design for systems in a normal form with multiple inputs and applied their approach to design a sliding mode controller for a 2 degree of freedom robot manipulator. The results show good tracking but considerable chattering. Fliess [21] proposed that for any general nonlinear system, state space coordinates exist so that the system can be represented in generalised controllable canonical form (GCCF). This particular form uses differentials of the outputs to define its state space. Sira-Ramirez [76, 77] proposed a sliding mode controller synthesis scheme based on GCCF. A particular way of designing the hyperplane was proposed. Recently Lu and Spurgeon [49] have proposed a generic strategy for sliding mode controller design based on nonlinear models in GCCF. These schemes are fairly generic as far as the modelling is concerned. The method uses differential input output models which represent a wide range of physical systems. For the sake of simplicity of notation, the
attention will be focussed on SISO systems. However, the scope of the design method is not limited to SISO systems only. Section 3.2 introduces the first design scheme which is illustrated through a design example in Section 3.3. The second design technique is discussed in Section 3.4 with the design example in Section 3.5. The two design techniques are compared in Section 3.6. The chapter is summarised in Section 3.7. The next section will describe the first controller design approach [47].

3.2 Indirect Sliding Mode Controller Design

The indirect sliding mode method is a new sliding mode approach based on differential input-output models. A particular choice of sliding surface is shown to effect asymptotic linearisation of the closed-loop system. This can be achieved with a chatter-free control signal as the resulting sliding mode control policy is dynamic in nature. The method will be seen to produce a procedure for controller construction which is more widely applicable than many traditional sliding mode approaches; essentially it circumvents the usual restrictions involving the need for the system of interest to be expressible in 'regular form'.

3.2.1 Model representation

A general nonlinear system can be represented by:

\[ y^{(n)} = \eta(\dot{y}, \dot{u}, t) \]  \hspace{1cm} (3.1)

where

\[ \dot{y} = (y, \dot{y}, ..., y^{(n-1)}) \]
\[ \dot{u} = [u, \dot{u}, ..., u^{(\alpha)}] \]

Here \( u^{\alpha} \) is the \( \alpha^{th} \) derivative of \( u \). This system can be written in the generalised controller canonical form (GCCF) [21].

\[ \dot{z}_1 = z_2 \]
Chapter 3. Sliding Mode Control in Nonlinear Systems

\[
\begin{align*}
\dot{z}_2 &= z_3 \\
& \vdots \\
\dot{z}_{n-1} &= z_n \\
\dot{z}_n &= \eta(z, \hat{u}, t) \\
\end{align*}
\]

(3.2)

where

\[
z = [z_1, z_2, ..., z_n]^T = [y, \dot{y}, ..., y^{(n-1)}]^T
\]

The associated zero dynamics are defined as:

\[
\eta(0, \hat{u}, t) = 0
\]

(3.3)

The nonlinear system (3.2) is called minimum phase if its zero dynamics are asymptotically stable.

### 3.2.2 Sliding Surface

For the indirect sliding mode method the sliding surface is proposed as:

\[
S = \sum_{k=1}^{n} a_k z_k + \eta(z, \hat{u}, t)
\]

(3.4)

where \(a_k\) are the parameters of an \(n^{th}\) order Hurwitz polynomial:

\[
\sum_{k=1}^{n+1} a_k \lambda^{k-1}
\]

When this sliding surface approaches zero, or when its magnitude becomes very small, then from (3.4):

\[
\eta(z, \hat{u}, t) = -\sum_{k=1}^{n} a_k z_k
\]

(3.5)

Hence from (3.2) the system dynamics in the ideal sliding mode can be rewritten as:

\[
\begin{align*}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= z_3 \\
& \vdots \\
\dot{z}_{n-1} &= z_n \\
\dot{z}_n &= -\sum_{k=1}^{n} a_k z_k
\end{align*}
\]

(3.6)
Thus the plant dynamics become equivalent to a time invariant linear system during the sliding mode. An asymptotic linearisation is effected. A nonlinear control must now be selected to ensure the sliding mode and hence the desired linear dynamics are exhibited.

### 3.2.3 Reachability condition

The condition to reach the sliding surface (reachability condition) is proposed as [46]:

\[
\dot{S} = -\gamma(\kappa, S) \tag{3.7}
\]

where for fixed design parameters

\[
\kappa = [\kappa_1, \kappa_2, ...]
\]

the following conditions are satisfied.

1. \(\gamma(\kappa, S)\) is continuous if \(S \neq 0\)
2. \(S\gamma(\kappa, S) > 0\) if \(S \neq 0\)
3. \(\gamma(\kappa, 0) = 0\)

For the SISO case the following reachability condition is used:

\[
\dot{S} = -(k_d\text{sign}(S) + k_cS) \tag{3.8}
\]

where \(k_d\) and \(k_c\) are positive parameters to be tuned during controller design. The \(\text{sign}(S)\) function is present in \(\gamma(S)\) because of its frequent use in the sliding mode literature where the associated robustness properties are well documented. The control \(\dot{u}\) satisfies:

\[
\{\dot{S} + k_d\text{sign}(S) + k_cS\} = 0 \tag{3.9}
\]

where the derivative of \(S\) is taken along the trajectories of equation (3.2). Having presented the theoretical formulation, an associated design algorithm is presented.
Chapter 3. Sliding Mode Control in Nonlinear Systems

3.2.4 Design Method

The design method may be decomposed into a number of steps which will be described in turn below:

Obtaining a Nonlinear Model.

The method for designing an indirect sliding mode controller for a SISO system will be explained. The first step is to obtain a representation of the system to be controlled in the form of equation (3.2). This goal can be achieved by three methods.

1. Assume that a nonlinear model is available in the form of differential input-output equations.

2. If a nonlinear model is in state space form

   \[ \dot{x} = f(x, u, t) \]
   \[ y = h(x, u, t) \quad (3.10) \]

   the state variable can be eliminated and the system transformed into differential input-output form assuming the system is observable [92].

3. If a differential input-output model is not available or a state space model (3.10) is difficult to express as a differential input-output model but input-output data is available then nonlinear identification methods can be employed to get a GCCF representation of the system. This is explained further in Chapters 4 and 5.

Specifying the Sliding Surface.

The dynamics of the sliding surface \( S \) are described by the roots of the Hurwitz polynomial given in equation (3.4). These roots determine the poles of the linear dynamics exhibited by the system (3.6) during the sliding mode. These dynamics should be close to those of the open loop plant otherwise a large controller effort may be required to force the nonlinear system to exhibit the desired linear dynamics.
Reachability Condition.

Having specified a sliding surface which defines the ideal linear dynamics, it is then necessary to select a nonlinear control to ensure that the sliding mode is attained. \( k_d \) and \( k_c \) are the design constants for specifying the reachability condition given in equation (3.9). These two constants act as weightings for the choice of the two modes of reachability i.e bang-bang control and asymptotic control. If the system is difficult to stabilise then \( k_d \) should be greater than \( k_c \). If smoothness of response is a priority then \( k_c \) should be increased.

3.2.5 Advantages of the Indirect Approach to Sliding Mode Control

Traditional sliding mode approaches usually employ a control independent sliding surface. It follows that dynamic sliding schemes only result when input-output systems which contain control derivatives are considered as in the work of Sira-Ramirez [78]. It should be noted that such dynamic sliding mode policies are desirable as they filter possibly discontinuous signals resulting in effective reduction of control chattering. Further, if the highest order derivative of the control appears nonlinearly in the input-output system, then it may be difficult to recover expressions for the control from the chosen reachability condition using traditional sliding mode approaches. Using the indirect approach, the sliding surface is always control dependent and therefore a dynamic controller which provides chattering reduction will result regardless of the particular input-output system representation. In addition, the highest order derivative of the control always appears linearly in the expression for \( \dot{S} \) which facilitates controller design. In the traditional sliding mode approach the system becomes equivalent to an \( n - 1 \) dimensional system when sliding. In the indirect approach, the sliding mode technique is used to asymptotically linearise the original nonlinear system; in the limit the sliding system thus becomes equivalent to an \( n \) dimensional linear asymptotically stable system.
3.3 Design Example: Underwater Vehicle Position Control

To illustrate the indirect sliding mode (ISM) design procedure, an ISM controller is designed for the position control of an underwater vehicle [85], which can be represented as:

\[ m\ddot{x} + c|\dot{x}| = u \quad (3.11) \]

Here \( x \) is the position of the underwater vehicle, \( m \) is the mass of the vehicle, \( c \) is the drag coefficient and the control input \( u \) is the force provided by a propeller. The main steps of the design process are:

3.3.1 Controller design

1. Assuming the following definitions:

\[ z_1 = x \]
\[ z_2 = \dot{x} \]
\[ \eta = \frac{-c}{m}z_2|z_2| + \frac{1}{m}u \]

The model can be written in GCCF form as:

\[ \dot{z}_1 = z_2 \]
\[ \dot{z}_2 = \frac{-c}{m}z_2|z_2| + \frac{1}{m}u \quad (3.12) \]

2. Choosing the sliding surface:

\[ S = a_1z_1 + z_2 + \eta(z, u) \]

where both \( a_1 \) and \( a_2 \) are fixed as 1.0, so that while sliding the system should have the dynamics of the Hurwitz polynomial [1 1].

3. Define the reachability condition as:

\[ \dot{S} = -(k_d\text{sign}(S) + k_cS) \quad (3.13) \]

The choice of the hard and soft gains will be discussed during simulation experiments.
4. Differentiating $S$ along the trajectory of equation (3.12).

$$\dot{S} = a_1z_2 + \eta(.) + \frac{(-c)}{m} \eta(.)[|z_2| + \text{sign}(z_2)] + \frac{1}{m} \ddot{u}$$

5. Substituting the value of $\dot{S}$ in equation (3.13) and solving for $\dot{u}$:

$$\dot{u} = m[-a_1z_2 - \eta(.) + \frac{c}{m} \eta(.)[|z_2| + \text{sign}(z_2)] - (k_d \text{sign}(S) + k_c S)]$$

3.3.2 Closed loop simulation

For simulation purposes, $\dot{u}$ will be integrated along with the derivatives of the other states. Therefore the control can be considered as an additional state for the purpose of integration. These values of the constants are used for simulation:

$$m = 1.0$$
$$c = 1.0$$

A traditional choice for the reachability parameters could be a high gain $k_d$ with zero soft gain $k_c$. The first simulation is performed with:

$$k_d = 60$$
$$k_c = 0$$

The states are shown in Figure 3.1. Figures 3.2 and 3.3 show the sliding surface and controller effort respectively. The effect of selecting a high gain for the discontinuous part is obvious. It has resulted in excessive chattering. Ideally speaking, there should be no chattering as the control policy is dynamic in nature. The chattering occurs because of the controller implementation procedure. As often is the case, the controller is implemented in discrete time, which results in discrete sampling of the control effort, thus producing a chattering effect. For this particular simulation, the step size is 10ms, which is particularly low for implementation purposes. In ISM controller design, the situation can be remedied by the proper choice of the hard and soft gains. The hard gain $k_d$ should be reduced and to ensure the speed of response the soft gain $k_c$ should have a high value.

The next simulation run is performed with:
Figure 3.1: States of the closed loop system with $k_c = 0$ and $k_d = 60$.

Figure 3.2: Sliding surface of the closed loop system with $k_c = 0$ and $k_d = 60$. 
Figure 3.3: Controller effort of the controller with $k_c = 0$ and $k_d = 60$.

Figure 3.4: States of the closed loop system with $k_c = 30$ and $k_d = 10$. 
\[ k_d = 10 \]
\[ k_c = 30 \]

The corresponding results are given in Figures 3.4 to 3.6. The chattering has been reduced due to the use of lower hard gain.

3.4 Direct Sliding Mode Control

3.4.1 Introduction

Sira-Ramirez [76] proposed a method for sliding mode controller design for SISO systems proposing a new kind of sliding surface design. Advancing the design further, Lu and Spurgeon [50] proposed a new MIMO robust nonlinear sliding mode controller design technique. This method is still based on nonlinear differential input output models, thus rendering itself applicable to a wide range of systems. Above all, this design method is robust against a wide range of uncertainties. A SISO version of this method is presented next.
3.4.2 System model with uncertainty

Consider the system described by equation (3.2). As expected in the real world, this model represents the actual system approximately. The resulting uncertain system can be expressed as:

\[ y^{(n)} = \eta(\hat{y}, \hat{u}, t) + \Delta(\hat{y}, t) \]  \hspace{1cm} (3.14)

where \( \Delta(\hat{y}, t) \) represents any uncertainty in the system. This uncertainty is assumed to be unknown but bounded. The assumed bounds on the uncertainty are:

\[ \Delta(\hat{y}, t) \leq \rho ||\hat{y}|| + l \]
\[ \rho \geq 0 \]
\[ l \geq 0 \]  \hspace{1cm} (3.15)

For reasons explained later, \( \rho \) and \( l \) will be referred to as hard and soft bounds. This uncertainty can come from various sources, such as modelling approximation, short and
Chapter 3. Sliding Mode Control in Nonlinear Systems

long term parameter variations, external disturbances, measurement noise etc. It is further assumed that [50]:

1. The system is minimum phase.

2. $\eta(\hat{y}, \hat{u}, t)$ is a $C^1$ function i.e its first derivative exists and is continuous.

3. $\eta(\hat{y}, \hat{u}, t)$ satisfies the regularity condition i.e

$$\frac{\partial(\eta(\hat{y}, \hat{u}, t))}{\partial(\hat{u}^o)} \neq 0$$

while assuming that all the elements of $\hat{u}$ are present. The system (3.15) can be rewritten as:

$$\begin{align*}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= z_3 \\
& \vdots \\
\dot{z}_{n-1} &= z_n \\
\dot{z}_n &= \eta(z, \hat{u}, t) + \Delta(\hat{y}, t)
\end{align*}$$

(3.16)

where all the notation is the same as in equation (3.2).

3.4.3 Direct Sliding Surface

The sliding surface for the above system is given as:

$$S = \sum_{k=1}^{n} a_k z_k$$

(3.17)

where $a_k$ are the parameters of an $n^{th}$ order Hurwitz polynomial:

$$\sum_{k=1}^{n+1} a_k \lambda^{k-1}$$

with

$$a_{n+1} = 1$$

This surface was proposed for the first time by Sira-Ramirez for single input systems [76]. When this sliding surface is close to zero, then from (3.17):

$$z_n = -\sum_{k=1}^{n-1} a_k z_k$$

(3.18)
Chapter 3.  Sliding Mode Control in Nonlinear Systems

Hence from (3.2) the reduced order system dynamics in the ideal sliding mode can be rewritten as:

\[
\begin{align*}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= z_3 \\
& \quad \vdots \\
\dot{z}_{n-1} &= -\sum_{k=1}^{n-1} a_k z_k \\
\end{align*}
\]

(3.19)

Thus the nonlinear plant dynamics become equivalent to a time invariant stable linear system during the sliding mode, whose dynamics can be specified by the design engineer.

A nonlinear control must now be designed so that the controller drives the system into the sliding mode and ensures the system remains confined to the desired reduced order linear dynamics. The reachability condition for this controller is defined in the same way as for the direct sliding mode controller (equation (3.7)). A systematic way for robust direct sliding mode (DSM) controller design is outlined in the next section. In the next section, it will be shown how to relate the bounds on the uncertainty to the determination of the reachability.

3.4.4 Design Method

The overall design procedure can be decomposed into a number of steps:

1. Choose the direct sliding surface as given in equation (3.17). The selection of the dynamics of the sliding mode is made as in the case of the indirect sliding mode (Section 3.2).

2. Estimate the bounds \( \rho \) and \( l \) on the uncertainty in equation (3.15). There are various ways to find out these bounds. Basically, these bounds can be estimated from the analysis of residues relating to the modelling error in the case of system identification. Then an estimation technique can be employed to fit a function to the residues. Alternatively, an envelope function can be estimated which can serve as the upper bound on the error. The next steps relate the uncertainty bounds \( \rho \) and \( l \) to \( k_c \) and \( k_d \).
3. Choose arbitrarily parameters $\theta$ and $\theta_o$ such that:

$$0 < \theta < 1$$

such that:

$$\theta + \theta_o = 1$$

This variable is mainly used for analysis purpose.

4. Compute $\rho_1$ as:

$$\rho_1 = \rho(1 + \max_{1 \leq k \leq n} |a_k|n)$$

where $\rho$ is the bound on the uncertainty defined before in equation (3.15), and $a_k$ are the coefficients of the Hurwitz polynomial determining the sliding surface.

5. Hence $\rho_2$ is computed as:

$$\rho_2 = \rho + \frac{(\rho_1)^2}{4\theta}$$

6. If $A$ is the companion matrix of the Hurwitz polynomial from equation (3.17) and $D \in \mathbb{R}^{n-1}$ is:

$$D = [0, 0, 0, ..., 1]^T$$

then the matrix $B$ is given as the solution of the Lyapunov equation:

$$A^T B + B A = -C$$

where $C$ is positive definite.

7. The lower bound on the soft gain $k_c$ (equation 3.8) is:

$$k_c - \frac{1}{\theta_o}[BD]^T[BD] + \rho_2 > 0$$

(3.20)

where $D \in \mathbb{R}^{n-1}$ and $B \in \mathbb{R}^{(n-1) \times (n-1)}$. Similarly the lower bound on $k_d$ is:

$$k_d > l$$

8. Differentiate $S$ along the trajectory of equation (3.17) to get $\dot{S}$.

9. Substitute the computed $\dot{S}$ in the equation 3.8 and solve for the highest derivative of input $u$. 
3.4.5 Advantages of the direct approach to sliding mode control

As mentioned previously, it has been proven that the DSM controller is robust against bounded uncertainties. Contrary to a traditional sliding mode controller, the control policy will be dynamic in nature if the model contains the derivatives of the control input.

3.5 Design Example: Direct sliding mode design for underwater vehicle

To illustrate the DSM controller design procedure the example considered in Section 3.3 will be used again. The model parameters are not assumed constant. Physically speaking, the approximate behaviour of the parameters varies as:

\[ m = 3 + 1.5 \sin(|z_2|t) \]
\[ c = 1.2 + 0.2 \sin(|z_2|t) \]

It is also known that the parameters are bounded as [86]:

\[ 1.0 \leq m \leq 5.0 \]
\[ 0.5 \leq c \leq 1.5 \]

This parameter variation causes uncertainty in the model which is represented by \( \Delta(.) \), as shown below:

\[ \dot{z}_1 = z_2 \]
\[ \dot{z}_2 = \frac{-c}{m} z_2 |z_2| + \frac{1}{m} u + \Delta(.) \quad (3.21) \]

It is necessary to find out the bounds \( \rho \) and \( l \) (equation 3.16) on this uncertainty due to the perturbations in \( m \) and \( c \).

3.5.1 Calculation of the uncertainty bounds

The function \( \eta(.) \) is defined as:

\[ \eta(z, u) = \frac{(-c)}{m} z_2 |z_2| + \frac{1}{m} u \]
while $\eta_{\text{per}}(.)$ is defined as:

$$\eta_{\text{per}}(.) = \eta(.) + \Delta(.)$$

Perturbing $m$ and $c$ in the expression for $\eta(.)$ to get an expression for $\eta_{\text{per}}(.)$:

$$\eta_{\text{per}}(z, u, t) = \frac{-(c + \delta c)}{m + \delta m} z_2|z_2| + \frac{1}{m + \delta m} u$$

Considering the first two terms of the binomial series of the denominators and simplifying:

$$\eta_{\text{per}}(z, u, t) = \eta(z, u, t) + \left(\frac{-1}{m}\right)(\delta c - \frac{\delta mc}{m} - \frac{\delta m \delta c}{m}) z_2|z_2| + \frac{\delta m}{m} u$$

If:

$$\delta b = \left(\delta c - \frac{\delta mc}{m} - \frac{\delta m \delta c}{m}\right)$$

Then:

$$\eta_{\text{per}}(z, u, t) = \eta(z, u, t) + \left(\frac{-1}{m}\right) \delta b z_2|z_2| + \frac{\delta m}{m} u$$

Considering the definition of $\eta_{\text{per}}(z, u, t)$:

$$\Delta(.) = \left(\frac{-1}{m}\right) \delta b z_2|z_2| + \frac{\delta m}{m} u$$

Taking norms of both sides:

$$\|\Delta(.)\| \leq \left|\left(\frac{-1}{m}\right) \delta b z_2\right| + \left|\frac{\delta m}{m} u\right|$$

Implying:

$$\|\Delta(.)\| \leq \left|\left(\frac{-1}{m}\right) \delta b \|z_2\|\|z\| + \left|\frac{\delta m}{m} u\right|$$

Comparing the above equation with the bounds given in equation (3.16), $\rho$ and $l$ are computed as:

$$\rho = \left|\left(\frac{-1}{m}\right) \delta b |z_2|\right|$$

and

$$l = \left|\frac{\delta m}{m} u\right|$$

In a physical system, the input signals are bounded. Suppose the input signal is bounded by $u_{\text{lim}}$. Similarly, it is realistic to assume that the velocity ($z_2$) of the underwater vehicle is also bounded by $z_{2\text{lim}}$. Hence, the bounds are:

$$\rho = \left|\left(\frac{-1}{m}\right) \delta b |z_{2\text{lim}}|\right|$$

and

$$l = \left|\frac{\delta m}{m} u_{\text{lim}}\right|$$
3.5.2 Controller design

The main steps in the design are:

1. The sliding surface is chosen as:

   \[ S = a_1 z_1 + z_2 \]

   with \( a_1 = 1.0 \) and \( a_2 = 1.0 \).

2. \( \theta_o \) and \( \theta \) are arbitrarily chosen as 0.01 and 0.99 respectively.

3. The reachability condition is as before

   \[ \dot{S} = -(k_d \text{sign}(S) + k_c S) \]  \hspace{1cm} (3.22)

   The gains are computed from the uncertainty bounds according to the procedure outlined before.

   \[ k_c = 20.25 \]

   \[ k_d = 15; \]

4. Differentiate \( S \) along the trajectory of equation (3.12):

   \[ \dot{S} = a_1 z_2 + \frac{-c}{m} \dot{z}_2 |z_2| + \frac{1}{m} u \]

5. Substitute \( \dot{S} \) in equation (3.22) and solve for \( u \):

   \[ u = m[-a_1 z_2 + \frac{c}{m} \dot{z}_2 |z_2| - (k_d \text{sign}(S) + k_c S)] \]

3.5.3 Simulation results

In the simulations the parameters are varied according to the given trajectories. The controller is thus tested in the presence of uncertainties to demonstrate the robustness. In the first simulation run the gains computed from the uncertainty bounds are used, resulting in some chattering. Since the bounds are quite conservative, they are relaxed (\( k_d \) is reduced to 1.0) and it is observed that the controller shows robust performance even for lower gains with reduced chattering. The conservatism in the Lyapunov analysis
Figure 3.7: States of the closed loop system with $k_c = 20.25$ and $k_d = 15.0$.

Figure 3.8: Sliding surface of the closed loop system with $k_c = 20.25$ and $k_d = 15.0$. 
Figure 3.9: Controller effort of the closed loop system with $k_c = 20.25$ and $k_d = 15.0$.

Figure 3.10: States of the closed loop system with $k_c = 20.25$ and $k_d = 1.0$. 
Figure 3.11: Sliding surface of the closed loop system with $k_c = 20.25$ and $k_d = 1.0$.

Figure 3.12: Controller effort of the closed loop system with $k_c = 20.25$ and $k_d = 1.0$. 
based design is a common problem. The chattering has occurred due to the fact that the derivatives of the control are not present in the model and DSM is used, resulting in a static controller. During the controller derivation, the function $\eta(.)$ in the system model (3.16) does not need to be differentiated, thus facilitating controller design.

### 3.6 Comparison

The two design techniques presented in this chapter are now compared:

1. Though the ISM method appears robust, the robustness of the method is not formally proved yet. On the other hand, the robustness of the DSM technique has been proven in [49].

2. During the ISM controller computation, while solving out for the highest derivative of control, the function $\eta(z, \hat{u}, t)$ in the equation 3.2 needs to be differentiated. Problems may arise if the derivative of the function $\eta(z, \hat{u}, t)$ does not exist, which is not the case for the DSM design.

3. Problems may arise during implementation if the solution of the highest derivative of control is singular. This problem can be encountered in both design techniques.

4. Since the design methods give the highest derivative of the control, the given expression needs to be integrated several times during the actual implementation of the controller. This may lead to the classical problem of integrator wind up.

5. In ISM design, the highest derivative of the control always appear explicitly during controller calculation, which is not the case for DSM design.

6. If the control derivatives are not present in the system model then the DSM control policy will not be dynamic in nature resulting in chattering of the control. ISM control policy is always dynamic, thus avoiding chattering. This fact can be seen in the design example of the underwater vehicle control. The system model does not have control derivatives. Consequently, the chattering in the DSM controller simulation is apparent, even for the lowered hard gain, unlike the ISM simulations.
where chattering is very low, which is mainly because of the discretisation of the controller.

3.7 Summary

This Chapter covers the nonlinear approaches to the sliding mode controller design. Particular attention is paid to two generic nonlinear model based sliding mode controller design techniques. These techniques were proposed by Lu and Spurgeon [49, 47, 50]. After presenting the two techniques, their explicit design procedures are explained. These are further illustrated through a design example relating to an underwater vehicle control problem. First Indirect Sliding Mode method is introduced. Issues like model representation, sliding surface design, the overall design method, the selection of different controller parameters and the advantages of this method are discussed. The effect of different gain selection on the controller performance is explored through closed loop simulation. Direct Sliding Mode is also introduced in detail. The design procedure is highlighted. Particular attention is paid to the design of the controller gains from the robustness margins which are eventually calculated from the bounds on the model uncertainty. A design example of the underwater vehicle is used to further explain the design procedure. For DSM controller design, uncertainty bounds of the underwater vehicle model arising from the parameter variation, are analytically computed. The model uncertainty bounds are thus calculated analytically and then used in DSM design method to compute the controller gains, hence ensuring a robust controller design. Robustness of the controller is shown by simulating the DSM controller on a model with uncertainties included. It is shown through the closed loop simulations that the gains computed from the quadratic stability results are quite conservative and cause excessive chattering. The conservatism is proven by simulating a lower gain controller giving good performance. At the end, both techniques are compared on the basis of design and implementation issues.

Though these methods are fairly generic, they suffer from the fact that the model has to be in GCCF form, which is not always the case, especially in industry, thus limiting the scope of application of these techniques. Therefore, it is needed to explore the nonlinear modelling techniques which use the plant input-output data to establish a
GCCF model of the plant. The next two chapters investigate various ways to find a suitable model for a given system in order to apply the above-mentioned techniques. A novel neural network-based modelling technique is proposed in Chapter 4. In Chapter 5, a new method to establish a GCCF model by employing the sliding mode concepts is given.
Chapter 4

Sliding Mode Control Based On Neural Networks

The controller design schemes proposed by Lu and Spurgeon [50, 47, 49] have been reviewed in the previous Chapter. But they have assumed that a nonlinear GCCF model is available, which is often not the case. There is a need for a nonlinear modelling technique which would be used to acquire such models. A new framework for nonlinear continuous time modelling using neural networks is proposed, it is also shown that the models obtained through this scheme are suitable for the controller design schemes mentioned above. This particular use of the neural networks in continuous time modelling and then in sliding mode controller design is new in itself. The neural network training algorithms like backpropagation method use quadratic cost function, similar to the one used in least square estimation methods. For least square estimators and maximum likelihood estimators Huber [62] proposed a cost function which is robust against Non-Gaussian noise. But so far no one has used this function in backpropagation algorithms. A modified backpropagation algorithm with Huber’s function is derived. The formulation of Huber’s cost function easily lends itself to the uncertainty bounds computation used in direct sliding mode controller design method. Including this method, other novel ways are also suggested to compute the uncertainty bounds from the neural network training data and neural network weights. The practicality of all these methods is demonstrated through design examples.
4.1 Introduction

As indicated in Chapter 3 a nonlinear identification method is needed to get a differential input/output model suitable for use in the dynamic sliding mode control schemes described in Chapter 3. If the plant is unknown but the order of the plant is roughly known, or if a state space model is known but it is difficult to get a differential input-output model from this, then it is necessary to identify such a model. It is obvious from equation (3.2) that the identification problem is concerned with identifying the following equation

\[ y^{(n)} = \eta(\dot{y}, \dot{u}, t) \]  

where \( n \) is the highest derivative of \( y \) in the system to be identified. Using the same notation as in equation 4.1 the goal is to find a nonlinear mapping which can transform \( \{\dot{u}, \ddot{y}, t\} \) to the corresponding \( y^n \). Cybenko [12] proved that any continuous mapping over a compact domain can be approximated as accurately as needed by a feed forward network with one hidden layer and sufficient number of neurons. A compact set in \( \mathbb{R}^n \) is a closed and bounded set. Hence it is possible to use neural nets to identify the desired nonlinear mapping \( \eta(.) \). Feed Forward Neural Networks (FFNN) , as described in Section 4.3 are used for the sliding mode control scheme, because their structure easily lends themselves for mathematical modelling and the manipulation needed for controller design, as discussed later.

After reviewing the relevant literature in Section 4.2, feedforward neural networks are introduced in Section 4.3. A robust backpropagation training method is proposed in Section 4.4. A new framework for sliding mode controller design based on neural networks is considered in Section 4.5 and illustrated through a design example in Section 4.6. Novel methods for the computation of uncertainty bounds from network training data are presented in Section 4.7. Finally the chapter is concluded in Section 4.8.

4.2 Literature review

In this section the literature is reviewed with two specific points of interest. People have used neural networks as a tool for nonlinear modelling with emphasis on control applications. The first subsection addresses this issue. The use of sliding mode theory
in conjunction with neural networks has been reported in various places. This is the topic of the second subsection.

4.2.1 Nonlinear identification using neural networks

The literature concerning only parametric identification is reviewed because a parametric model is needed for sliding mode controller design. Industrial processes have been identified using neural networks. Some authors have used feedforward neural networks (FFNN) for this purpose. MacMurray and Himmelbau [51] trained a FFNN for a distillation column. They used this nonlinear model in a Model Predictive Control Scheme. Chessari and associates [9] used FFNN to identify a distillation column, hence finding unknown functions in a known model structure. This model was incorporated in the differential algebra based nonlinear control method of input output linearisation. Kuschewski and Zak [43] trained a FFNN to model an inverted pendulum and hence used it in adaptive control.

Various authors have used Radial Basis Function Networks (RBF). Seborg [71] employed RBF networks for pH neutralisation modelling for an Internal Model Control scheme. Chu and associates [10] also used RBF for continuous nonlinear system modelling.

A few authors have preferred Voltera series functions. Davies and Goodhart [13] identified a gas fired furnace with Voltera series for controlling the furnace with Model Predictive Control. Holt and co-workers [31] presented an interesting comparison of the nonlinear modelling performance among FFNN with linear terms, regression and genetic algorithms. They concluded that FFNN did not outclass other estimation methods for nonlinear modelling. However, this is an inference based on one particular modelling experiment which cannot be generalised. Recurrent neural networks (RNN) have found wide applications in discrete time modelling. Recurrent neural networks are the networks with feedback connections from the output layer to the input layer (with or without delay). Due to the feedback connections, they exhibit hysteresis properties which are needed for dynamical system modelling. Seidl and his coworkers [72] showed that RNN can be used to model a wide class of dynamic discrete time systems. Yu [106] demonstrated the use of both FFNN and recurrent neural networks for modelling discrete time systems. He showed some successful simulation results. Jubien et al [34]
presented a training procedure for RNN, based on a gradient method. Ku and Kwang [42] proposed a new RNN structure called a diagonal recurrent neural network for discrete time system modelling and control.

It is seen that neural networks have been extensively used for nonlinear modelling in a wide range of applications. However, a major shortcoming in the reported applications is the fact that in most of these cases the models obtained are discrete time models. Being nonlinear, these models cannot be converted into their continuous time counterparts in a generic way. Hence, the discrete time models are not suitable for the controller design schemes presented in Chapter 3.

In this chapter it will be attempted to use feedforward neural networks (FFNN) for continuous time model estimation. The various types of networks mentioned are discussed in detail in [11, 29]. A generalised representation of these networks is proposed by Sjoberg [81]. A review of the use of neural networks in control theory can be found in [97]. FFNN are chosen because their formulation is most suitable for the sliding mode control scheme.

4.2.2 Use of neural networks in variable structure control

Sliding mode control and neural networks have been used together in various, diverse ways. In some applications, neural networks are directly used for controller synthesis. The networks are used to approximate complex transforms. They are also employed as feedforward compensators to reduce chattering. A brief account of each is given.

Cao et al [8] proposed a novel approach for using neural networks for sliding mode controller synthesis. They used neural networks to determine a stable switching surface for a given nonlinear system. Drakunov and associates [16] used neural networks for approximating a complex transform. They designed a controller for a beam with piezoelectric actuation. The original system is in distributed parameter form. It is converted to a second order partial differential equation form using an integral transform. This integral transform is difficult to compute analytically and is hence modelled by neural networks. For the transformed system a variable structure control is designed.

Xu and his coworkers [102] have reported another application of neural networks for related controller synthesis. VSC is applied to a feedback linearised second order non-
linear system resulting in high chattering. A low pass filter is used to get the average control signal. A neural net is trained to predict the average control from the system parameters. Then this network is used as a feedforward compensator to reduce chattering.

Sira-Ramirez and Zak [107, 80] used variable structure control to modify the training algorithm (Widrow Hoff Learning Rule) for neural networks. The modified algorithm can be used for on line training. Sabanovic et al [69] applied neural network to approximate the unknown part of the equivalent control in a discrete time sliding mode control scheme. Fu [25] used neural networks to identify a linear system and then used it in a variable structure based adaptive tracking scheme. Won and coworkers [100] designed a dynamic sliding mode controller from the analytic model of an air to fuel ratio mechanism in a spark ignition engine and used neural networks to estimate the one dimensional nonlinear characteristic of the air mass flow rate sensor, which is one of the measurements for the control scheme. The next section serves as an introduction to FFNN.

### 4.3 Formulation of feedforward neural networks

This Section is essentially an introduction to the feedforward neural networks. A detailed introduction can be found in the textbooks of neural networks like Haykin's book [29]. The introduction is provided so that the reader would become familiarised with the mathematical modelling of the feedforward neural networks which will be utilised in the later Sections of this Chapter.

The basic building block of a FFNN is a neuron. A neuron can be described as a non-linear mapping \( \mathbb{R}^{l_0} \to \mathbb{R} \). \( l_0 \) is the number of inputs to the neuron. The output of the neuron is

\[
p = \Psi(\sum_{j=1}^{l_0} w_j q_j + b_o)
\]

where \( w_j \) is a scalar being multiplied by the corresponding input \( q_j \), \( b_o \) is a constant scalar called bias. \( \Psi(.) \) is a monotonic function whose precise structure will be explained later. If \( \bar{w} = (w_1, w_2, ... w_{l_0})^T \) and \( \bar{q} = (q_1, q_2, ... q_{l_o})^T \) then the above equation may be
written in vector form as:
\[ p = \Psi(w^T \bar{q} + b) \]

Now consider a layer of \( l_1 \) neurons i.e. a set of \( l_1 \) neurons receiving the same input \( \bar{q} \). Then the output of this layer \( p_1 \in \mathbb{R}^{l_1} \) is
\[ p_1 = \Psi(W_1 \bar{q} + b_1) \]

with \( W_1 \in \mathbb{R}^{l_1 \times l_0} \) and \( b_1 \in \mathbb{R}^{l_1} \). The above equation formulates a single layer FFNN with \( l_0 \) inputs and \( l_1 \) outputs. This can be extended easily to a two layer FFNN. Take \( l_2 \) number of neurons in the second layer. In a two layer FFNN all the outputs of the first layer are connected to the inputs of the second layer weighted by a set of weights described by the matrix \( W_2 \in \mathbb{R}^{l_2 \times l_1} \). The output \( p_2 \in \mathbb{R}^{l_2} \) of a two layer network is
\[ p_2 = \Psi(W_2(\Psi(W_1 \bar{q} + b_1) + b_2)) \] (4.2)

where \( b_2 \in \mathbb{R}^{l_2} \). The same formulation can be extended to networks with a higher number of layers.

There are three choices of \( \Psi(.) \) (usually known as squashing functions) most frequently used [1].

1. Sigmoidal function given as
\[ \Psi(x) = \frac{1}{1 + e^{-x}} \]
   This function maps \( \mathbb{R} \rightarrow (0, 1) \).

2. Hyperbolic tangent function. \( Tanh(.) \) function maps \( \mathbb{R} \rightarrow (-1, 1) \).

3. The third function often used is a linear function where it is assumed that there is a linear relationship between the input and output.

After determining the structure, the next step is to train the network.

The FFNN is required to follow a prescribed output trajectory called the target output (for a given set of inputs). The difference between the actual net output and the target output constitutes a cost function. The minimisation of this function with respect to the network weights is called Network Training. Several optimisation methods are used for this purpose. The backpropagation method is most frequently used. For details of this method see Krogh [41]. In the next section a backpropagation algorithm with a robust cost function is presented.
4.4 Backpropagation training algorithm with robust cost function

Traditionally, a quadratic cost function is used in the backpropagation algorithm. This trend has its roots in the parameter estimation techniques like least square estimation. Huber [62] showed that a quadratic cost function is only suitable for least square parameter estimation applied on the data corrupted with the Gaussian noise. He proposed another cost function called Huber’s function which is robust against non-gaussian noise as well. This function will now be adopted in the backpropagation algorithm and the whole algorithm is derived again.

The backpropagation algorithm with Huber’s function will be presented now. For the sake of simplicity, bias is considered as another weight with unit input. Throughout this section \( u_i^{[n]} \) means the input to the squashing function of the \( i^{th} \) neuron in the \( n^{th} \) layer. Similarly \( p_i^{[n]} \) is the output of the squashing function of the \( i^{th} \) neuron in the \( n^{th} \) layer. Also \( p_i^{[0]} \) is the vector containing all outputs of the \( l^{th} \) layer. \( e_i \) is the output error of the \( i^{th} \) neuron in the output layer. \( u^{[0]} \) is the input vector going into the network. The extended three layer version of the network (equation (4.2)) is:

\[
p^{[3]} = \Psi[W_3\Psi [W_2\{\Psi(W_1 u^{[0]} + b_1) + b_2}\}] + b_3
\]

In the neural network literature the most commonly used cost function for optimisation is:

\[
E = \sum_{i=1}^{l_3} \sigma_s(e_i)
\]  

(4.4)

where:

\[
\sigma_s = \frac{1}{2} e_i^2
\]

and

\[
e_i = d_i - p_i^{[3]}
\]

where \( d_i \) is the target output for \( i^{th} \) neuron in the output layer (3rd layer) and \( p_i^{[3]} \) is its actual output. This cost function is good for data corrupted with Gaussian noise, but it does not show good performance for non-Gaussian noise. Huber suggested another function which is also suitable for non-Gaussian noise [62]. This can be given as:

\[
E = \sum_{i=1}^{l_3} \sigma_h(e_i)
\]

(4.5)
where:

\[ \sigma_h = \frac{1}{2} |e_i|, \text{if } |e_i| \leq \beta, \]
\[ = \beta |e_i| - \frac{\beta^2}{2}, \text{if } |e_i| > \beta, \] (4.6)

with \( \beta \) a positive constant. The weight update \( \Delta w_{ij}^{[3]} \) in the third layer is given as:

\[ \Delta w_{ij}^{[3]} = -m \frac{\partial E}{\partial w_{ij}} \]

Using the chain rule to avoid direct differentiation of the output error cost function with the weights:

\[ \Delta w_{ij}^{[3]} = -m \frac{\partial E}{\partial u_i^{[3]}} \frac{\partial u_i^{[3]}}{\partial w_{ij}^{[3]}} \]

Defining \( u_i^{[3]} \) in terms of \( w_{ij}^{[3]} \) and \( p_i^{[2]} \):

\[ u_i^{[3]} = \sum_{j=1}^{n^3} w_{ij}^{[3]} p_i^{[2]} \]

A new variable \( \delta_i^{[3]} \) is introduced to represent the partial derivative of the cost function \( E \) with respect to \( u_i^{[3]} \):

\[ \delta_i^{[3]} = -\frac{\partial E}{\partial u_i^{[3]}} \]
\[ \delta_i^{[3]} = -\frac{\partial E}{\partial e_i^{[3]}} \frac{\partial e_i^{[3]}}{\partial u_i^{[3]}} \]

From the definitions of \( E \) and \( e_i \) in equation 4.6 it follows:

\[ \delta_i^{[3]} = e_i \frac{\partial p_i^{[3]}}{\partial u_i^{[3]}}, \text{if } |e_i| \leq \beta \]
\[ = \beta \text{sign}(e_i) \frac{\partial p_i^{[3]}}{\partial u_i^{[3]}}, \text{if } |e_i| > \beta \] (4.7)

Hence the weight update for the third layer is:

\[ \Delta w_{ij}^{[3]} = m \delta_i^{[3]} p_i^{[2]} \]
The weight update for the second layer is computed next. Starting from the partial derivative of the cost function with respect to the weights in the second layer:

\[ \Delta w_{ij}^{[2]} = -m \frac{\partial E}{\partial w_{ij}^{[2]}} \]

\[ \Delta w_{ij}^{[2]} = -m \frac{\partial E}{\partial u_i^{[2]}} \frac{\partial u_i^{[2]}}{\partial w_{ij}^{[2]}} \]

\[ \delta_i^{[2]} \] for the second layer is introduced in a similar way as done for the third layer:

\[ \delta_i^{[2]} = -\frac{\partial E}{\partial u_i^{[2]}} \]

\[ \delta_i^{[2]} = -\frac{\partial E}{\partial p_i^{[2]}} \frac{\partial p_i^{[2]}}{\partial u_i^{[2]}} \]

It will be attempted to expand the second layer "delta term" and get its equivalent in the known terms of the third layer.

\[ \frac{\partial E}{\partial p_i^{[2]}} = -\sum_{j=1}^{n_3} \frac{\partial E}{\partial u_j^{[3]}} \frac{\partial u_j^{[3]}}{\partial p_i^{[2]}} \]

\[ \frac{\partial E}{\partial p_i^{[2]}} = \sum_{j=1}^{n_3} \left[ -\frac{\partial E}{\partial u_j^{[3]}} \frac{\partial}{\partial p_i^{[2]}} \left[ \sum_{k=1}^{n_2} w_{i[k]}^{[3]} p_{k} \right] \right] \]

\[ \frac{\partial E}{\partial p_i^{[2]}} = \sum_{i=1}^{n_3} \delta_i^{[2]} u_i^{[3]} \]

Substituting this value in the definition of \( \delta_i^{[2]} \):

\[ \delta_i^{[2]} = \frac{\partial p_i^{[2]}}{\partial u_i^{[2]}} \sum_{i=1}^{n_3} \delta_i^{[3]} w_i^{[3]} \]

Hence the weight update in the second layer is:

\[ \Delta w_{ij}^{[2]} = m \delta_i^{[2]} p_{i}^{[1]} \]

Similarly for the first layer:

\[ \Delta w_{ij}^{[1]} = m \delta_i^{[1]} u_{i}^{[0]} \]

\[ \delta_i^{[1]} = \frac{\partial p_i^{[1]}}{\partial u_i^{[1]}} \sum_{i=1}^{n_2} \delta_i^{[2]} w_i^{[2]} \]
Therefore, the weight updates obtained can be used for network training.

4.5 Combination of the sliding mode controller and neural network model

This section explains the combination of sliding mode control schemes introduced in Chapter 3 with a neural net model. This method is useful in the following two cases:

1. The plant is unknown, but input-output information is available and the system is internally stable. This is the case for many industrial plants. The input-output data \((\hat{y}, \hat{u})\) may be used for training the neural model.

2. A state space model

\[
\begin{align*}
\dot{x} &= f(x, u, t) \\
\Sigma: y &= h(x, u, t)
\end{align*}
\]  

(4.8)

is known, which is observable, but which is difficult to eliminate into an input-output model. From (4.8) the following can be calculated:

- (a) the relative order
- (b) \(y, \dot{y}, ..., y^{(n-1)}\)

From these relations, the data for \((\hat{u}, \hat{y})\) can be obtained which can be used to train a FFNN model of the form:

\[
y^{(n)} = \eta(\hat{y}, \hat{u}, t)
\]

4.5.1 Initial condition issues

For using a neural net model to identify a dynamical plant or a differential input-output model the initial conditions must be fixed. This means that a neural model practically represents an input-output mapping which corresponds to the fixed initial conditions \(x_{t=0} = x_o\). In the practical control process, if an initial condition \(x_o^*\) slightly different from \(x_o\) is used, then it will cause a mismatch between the FFNN and the
corresponding model (4.8) or unknown plant. In general the input-output model is subjected to uncertainties as:

\[ y^{(n)} = \eta(\hat{y}, \hat{u}, t) + \Delta_1(t) + \Delta_2(t) \]

where \( \Delta_1(t) \) is caused by the mismatch between the neural model and the plant data when both are subject to the initial conditions used for training. \( \Delta_2(t) \) is caused by the mismatch between the neural model corresponding to \( x_0 \) and the plant with different initial conditions. By employing a sliding mode philosophy, with the attendant robustness properties, it is hoped to maintain robust control despite the presence of the uncertain elements \( \Delta_1(t) \) and \( \Delta_2(t) \). The next section illustrates the indirect sliding mode design procedure with a neural network based model.

4.5.2 Structural issues

So far it is clear that various training methods exist to find the suitable weights of a neural network for function approximation. However, there are no clear cut methods to decide upon the structure of the network itself i.e to decide upon the number of hidden layers, number of inputs and number of neurons in each layer. A trial and error method called Network Pruning is often suggested. The method consists of training a very large network initially and then systematically pruning out the redundant neurons [11]. Rhodes and Morari [64] used a method to determine the order of a discrete time model directly from the system input output data.

Once a neural network has been used to get a model of the form:

\[ y^{(n)} = \eta_{nn}(\hat{y}, \hat{u}, t) \] (4.9)

Then this model can be transformed into:

\[
\begin{align*}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= z_3 \\
& \vdots \\
\dot{z}_{n-1} &= z_n \\
\dot{z}_n &= \eta_{nn}(z, \hat{u}, t)
\end{align*}
\] (4.10)
which is a GCCF form and hence useful for sliding mode controller design. The next section illustrates the whole procedure through a design example.

4.6 Design Example: Output tracking for a Tunnel Diode circuit using indirect sliding mode with neural model

An indirect sliding mode controller for a tunnel diode circuit [37] based on a neural network model is presented. The circuit is shown in Figure (4.1). The nonlinear dynamics are well known and the system may be described as:

\[
\begin{align*}
\dot{x}_1 &= \frac{1}{C}[-g(x_1) + x_2] \\
\dot{x}_2 &= \frac{1}{L}[-x_1 - Rx_2 + u] \\
g(x_1) &= \sin(x_1) \\
y &= x_1
\end{align*}
\] (4.11)

Using the same notation as in equation (3.2):

\[ z_1 = x_1 \]
\[ z_2 = \dot{x}_1 \]

A neural network is trained to approximate the function \( \eta(.) \) (see equation 4.1); in this way a GCCF approximation to equation (4.11) is developed. A sinusoidal signal of varying frequency is chosen for collecting training data. The system (4.11) is simulated with this input excitation. The system data thus obtained is used for network training. The second derivative of \( z_1 \) is selected as the target output for the network. Using the following network input:

\[
\begin{bmatrix}
u
z_1
z_2
\end{bmatrix}
\]

and \( \hat{z} \) as the target output for training, a neural model is identified. Training was done through Levenberg-Marquardt algorithm which is quasi Newton version of the backpropagation training method. The input for this experiment should have all the frequencies which can excite the system. A sinusoidal input of varying frequency would
serve the purpose. Hence, a chirp signal was selected to gather training input output data. Through extensive training experiments it was found that only one hidden layer with 5 neurons is sufficient to get a good mapping.

\[
\eta(u, z_1, z_2) = W_2(W_1 u_{nn} + b_1) + b_2
\]

\[
\Rightarrow \dot{z} = W_2(W_1 u_{nn} + b_1) + b_2
\]

(4.12)

where \(W_1 \in \mathbb{R}^{3 \times 3}, W_2 \in \mathbb{R}^{1 \times 3}, b_1 \in \mathbb{R}^3\) and \(b_2 \in \mathbb{R}\). The network is a double layer net with three inputs and one output which is the value of the function \(\eta(u, z_1, z_2)\) itself. The first layer contains 5 neurons and there is only one neuron in the second layer. The comparison of the actual system output and the neural model output for different input signals is shown in Figures (4.2) and (4.3). The GCCF representation of the system (4.11) is:

\[
\begin{align*}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= W_2(W_1 u_{nn} + b_1) + b_2
\end{align*}
\]

(4.13)
Using equation (3.4) the sliding surface is given as:

$$S = a_1z_1 + a_2z_2 + \eta(u, z_1, z_2)$$

Hence

$$\dot{S} = a_1\dot{z}_1 + a_2\dot{z}_2 + \dot{\eta}(.).$$

$\Rightarrow \dot{S} = a_1z_2 + a_2\eta(.) + \dot{\eta}(.)$ (4.14)

The derivative of $\eta(u, x_1, x_2)$ is needed for computing the controller.

$$\dot{\eta}(.) = \eta_u(.)\dot{u} + \eta_{z_1}(.)\dot{z}_1 + \eta_{z_2}(.)\dot{z}_2$$ (4.15)

If the weight matrix $W_1$ is given in vector form as:

$$W_1 = [w_1, w_2, w_3]$$

then

$$\eta_u = W_2 * Diag[w_1]$$

$$\eta_{z_1} = W_2 * Diag[w_2]$$

$$\eta_{z_2} = W_2 * Diag[w_3]$$

where $Diag[w_1]$ is a diagonal matrix with the vector $w_1$ as its principal diagonal. The control is computed from equation (3.9):

$$\rho = a_1\dot{z}_2 + a_2\dot{\eta}(.) + \eta_{z_1}(.)z_2 + \eta_{z_2}(.)\eta(.)$$

$$\dot{u} = -\frac{\rho + k_1\text{sign}S + k_2S}{\dot{\eta}_u(.)}$$

with the following controller parameters:

$$k_1 = 10$$

$$k_2 = 5$$

$$a_1 = 1$$

$$a_2 = 2$$

The first simulation is performed with the same initial conditions used for training as shown in Figure (4.4). The system states correspond to the actual system (4.11). There
Figure 4.4: The performance of neural network based ISM controller on the nominal nonlinear model (15) with the same initial conditions used for training
Figure 4.5: The performance of neural network based ISM controller on the nominal nonlinear model (15) with same initial conditions and only dynamic gain
is a small amount of chattering in the first simulation. This is due to an unnecessarily high discontinuous gain. The same simulation is performed again with the same parameters except for $k_1 = 0$. The results in Figure (4.5) show no chattering. A further simulation considers the effect of changes in the initial conditions (Figure (4.6)). It is noted that this design method is robust with respect to the uncertainty caused by the mismatch in identification and the uncertainty caused by using different initial conditions.

![Graphs showing State Trajectory, Controller Effort, State 2 Trajectory, and Sliding Surface](image)

Figure 4.6: The performance of Neural Network Based ISM Controller on the Nominal Nonlinear Model (15) with different initial conditions

The design of neural network model based ISM controller has been illustrated through a design example. In the next section neural model based DSM controller design is presented.
Chapter 4. Sliding Mode Control Based On Neural Networks

4.7 Determination of robustness bounds for neural network based direct sliding mode controller

A robust sliding mode controller design method was presented in Section 3.2 where it was shown that the controller is robust against bounded uncertainties arising from the system representation in GCCF form. This section deals with the estimation of the uncertainty bounds due to the neural modelling procedure. These estimation methods and the relevant DSM controller design are illustrated with design examples. In the last section (equation 4.16) it was shown that $\eta(.)$ can be estimated by training a neural network model such that:

$$\hat{\eta}(.) = \eta_{nn}(.) + \Delta(.)$$

where $\Delta(.)$ denotes the uncertainty caused by the network training error. Hence equation (4.16) becomes:

$$\begin{align*}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= z_3 \\
&\vdots \\
\dot{z}_{n-1} &= z_n \\
\dot{z}_n &= \eta_{nn}(z, \dot{u}, t) + \Delta(.) \tag{4.16}
\end{align*}$$

Comparing this equation with equation (3.16), a robust controller can be designed as long as $\Delta(.)$ is bounded by:

$$\|\Delta(.)\| < \rho \|z\| + l$$

The aim of this section is to investigate various ways to determine these robustness bounds in order to design a direct sliding mode controller based on the neural network model (4.16) and explicit uncertainty bounds relating to the model.

4.7.1 Uncertainty bound estimation using curve fitting

Alternatively, the uncertainty bounds can be obtained by fitting a function on the residues obtained from neural network training. The method will be illustrated by using tunnel diode circuit model used before (4.11). A neural network is trained to
obtain a model in the GCCF form in the similar way as before. The network used is a two layered neural network trained for tracking a demand of:

\[ x_1 = 0.5 \]

\[ x_2 = \sin(0.5) \]

The neural network based model is:

\[ \dot{z}_1 = z_2 \]

\[ \dot{z}_2 = W_2(W_1 u_{nn} + b_1) + b_2 \]  \hspace{1cm} (4.17)

The open loop system is simulated with different initial conditions of \([5.0, 5.0]\) (the nominal initial conditions used for training were \([1.0, 1.0]\)). The error data thus obtained from the two trajectories is used for determining the error bounds for the neural model. The norm of the error due to initial conditions perturbation is fitted against the norm of the state vector via a first order polynomial, such that:

\[ ||e|| = p_1 ||x|| + p_2 \]

This fitting is shown in the Figure 4.7. From the method of robust sliding mode controller design, the corresponding \(p\) and \(l\) are found such that:

\[ ||e|| < p ||x|| + l \]

where

\[ p_1 = 0.5709 \]

\[ p_2 = -0.344 \]

The sliding surface for the direct sliding mode controller design (DSM) is:

\[ s = a_1 x_1 + a_2 x_2 \]

\(a_1\) and \(a_2\) constitute a Hurwitz polynomial (\(a_1 = 1.0\) and \(a_2 = 2.0\)). The equation pertaining to the reachability condition is:

\[ \dot{s} = -(k_v s + k_d \text{sign}(s)) \]
Chapter 4. Sliding Mode Control Based On Neural Networks

Using this reachability condition the control is computed. The corresponding lower bounds for the soft \((k_c)\) and hard gain \((k_d)\) are computed by solving the corresponding Lyapunov equation as proposed in the Section 3.4.

\[
lb_{k_c} = 1.076
\]

\[
lb_{k_d} = 0.0
\]

The results for different gains are shown in the Figure 4.8. The continuous controller gain is \(k_c\) and the discontinuous gain is \(k_d\). As can be seen from the Figure 4.8, the system is unstable for \(k_c\) of 0.1 as the lower bound on \(k_c\) to keep the system stable is computed as 1.076. The performance improves as the gain is increased beyond \(\rho\).

### 4.7.2 Uncertainty bound estimation from Huber’s function

In Section 4.4, the backpropagation algorithm using Huber’s function (equation (4.6)) as the error function has been presented. This error function itself can be used to estimate the error bounds mentioned in equation (3.16) in Chapter 3. It is needed to estimate
Chapter 4. Sliding Mode Control Based On Neural Networks

State 1 Trajectory for $k_c=0.1 \ k_d=0$ State 2 Trajectory

Controller Effort Sliding Surface

State 1 Trajectory for $k_c=0.6 \ k_d=0$ State 2 Trajectory

Controller Effort Sliding Surface
Figure 4.8: Results
bounds for the uncertainty $\Delta(z,t)$ arising from the network training procedure. The network estimation error can be given as:

$$e = \hat{z}_n - \eta_{nn}(.)$$

But this error can also be incorporated in Huber’s error function (4.6):

$$\sigma_h = \beta|e| - \frac{\beta^2}{2}$$

Considering the case when $|e| > \beta$.

As discussed before the hard gain $k_d$ (responsible for chattering) is bounded from below by $l$. Consequently, it is preferable to have $l$ as low as possible. In the current scheme, this can be achieved by minimising $\beta$ against network weights. If the backpropagation algorithm is run with overall $\beta$ minimisation the chattering can be reduced. The backpropagation training procedure will give a measure of this Huber’s function which can be used to calculate the uncertainty bounds. It follows from the definition of Huber’s function:

$$|e| = \frac{\sigma_h}{\beta} + \frac{\beta}{2}$$

Which implies:

$$|e| \leq \frac{\sigma_{h_{max}}}{\beta} + \frac{\beta}{2}$$

Comparing the above inequality with the uncertainty bounds in equation (3.16):

$$\rho = \frac{\sigma_{h_{max}}}{\beta||z||}$$

and

$$l = \frac{\beta}{2}$$

When the training procedure from Section 4.4 is used to minimise both the output error and $\beta$ with respect to the weight of the networks, uncertainty bounds can be computed from this minimised $\beta$.

The tunnel diode example is used again for the verification of these bounds. The training procedure is the backpropagation algorithm with Huber’s cost function. The bisection algorithm is used to minimise error residues resulting from backpropagation method against $\beta$, as shown in figure 4.9. $\beta$ turns out to be 4.99. The bounds are estimated as:
As seen in the last subsection that the system is unstable for $k_c = 0.1$ and $k_d = 0$ which is lower than the bounds computed here. This implies that the bounds are conservative but "safe". This is a problem often encountered in Lyapunov based designs.

### 4.7.3 Uncertainty bound estimation using network distance

This subsection considers a bound estimation approach which is similar to the simultaneous stabilisation problem in the linear control literature. If two models $G_1$ and $G_2$ exist for two different setpoints in the envelope of operation of a system, then it is possible to guarantee the stability of the controller designed using $G_1$ when applied to $G_2$. Through network training, $\eta_1$ and $\eta_2$ are the two neural network models obtained for the GCCF forms of $G_1$ and $G_2$. If a DSM controller is designed with $G_1$ then it would be stable with $G_2$ if:

$$\Delta \eta(.) < \rho \|z\| + l$$
Chapter 4. Sliding Mode Control Based On Neural Networks

where

\[ \Delta \eta(.) = \eta_1 - \eta_2 \]

For linear squashing functions:

\[ \eta_1(.) = W_2^2[W_1^1 + B_1^1] + B_2^1 \]

and

\[ \eta_2(.) = W_2^2[W_1^2 + B_1^2] + B_2^2 \]

If \( \Delta W_i = W_{i}^2 - W_{i}^1 \) and \( \Delta B_i = B_{i}^2 - B_{i}^1 \), then with some algebra it can be shown that:

\[ \Delta \eta(.) = [W_{2}^2 \Delta W_1 + \Delta W_2 W_1^1 + \Delta W_2 \Delta W_1]x + W_{2}^1 \Delta B_1 + \Delta B_2 + \Delta W_2 \Delta B_1 + \Delta W_2 B_1 \]

Which implies:

\[ ||\Delta \eta(.)|| < \rho_w ||z|| + l_w \]

Where:

\[ \rho_w = W_2^1 \Delta W_1 + \Delta W_2 W_1^1 + \Delta W_2 \Delta W_1 \]

\[ l_w = W_2^1 \Delta B_1 + \Delta B_2 + \Delta W_2 \Delta B_1 + \Delta W_2 B_1 \]

The method is illustrated through a design example. The tunnel diode system given in equation (4.11) with parameters \( R = C = L = 1.0 \) is simulated in open loop and data obtained is used for network training. Then the system (4.11) is simulated with new parameter values of \( R = C = L = 5.0 \) and \( g(x_1) = x_1^2 \) with the same initial conditions as before. A second network is trained with this set of data but using the same weight initialisation as used for the previous training. Using these two sets of weights and biases, bounds are calculated from equations (4.18) and (4.19). These bounds are used in direct sliding mode controller design. The simulation results are shown in figures 4.10 and 4.11. The controller performance with the perturbed model is stable. The steady state error can be removed by using higher gains as shown in figure 4.12.

4.7.4 Statistical estimation of the uncertainty bound

A statistical bound on the network approximation error is proposed in [30]. The following notation is used. If \( M \) is the total number of hidden neurons, \( N \) is the number of
Chapter 4. Sliding Mode Control Based On Neural Networks

State 1 Trajectory for kc= 2.664 kd=1.163

Controller Effort

Figure 4.10: DSM controller with nominal system

State 1 Trajectory for kc= 2.664 kd=1.163

Controller Effort

Figure 4.11: DSM controller with perturbed system
training examples, \( p \) is the number of inputs, \( C_f \) denotes the first absolute moment of Fourier magnitude distribution of the target function \( f(.) \), and \( O(.) \) denotes the order of the argument then the mean integrated approximation error \( R \) is:

\[
R < O\left(\frac{C_f^2}{M}\right) + O\left(\frac{M_p}{N \log N}\right)
\]

There are two problems with this approach. The target function in the case of black box modelling is unknown and hence \( C_f \) cannot be computed. Secondly, this method will give a constant bound representing a worst case scenario. This high constant bound can only be used to calculate the lower bound on the hard gain, thus resulting in a high value of the hard gain. Since a higher hard gain will cause chattering, this method will not give a bound useful for good performance.

4.8 Summary

This Chapter has looked into the issue of nonlinear modelling of a system for robust sliding mode controller design. A novel way is suggested to employ neural networks for this purpose. This is extended further by exploring new ways to find the uncertainty
bounds associated with the neural network modelling. Various new methods to estimate uncertainty bounds are proposed. The structure and the weights of the neural networks is analysed to get such bounds. Alternately, it is shown that residuals from neural network training can be utilised for uncertainty bounds estimation. Huber's function is frequently used in least square type parameter estimation techniques. For the first time it is incorporated in backpropagation theorem. It is also shown that the structure of the Huber's function can be exploited to determine the uncertainty bounds, and in turn, the sliding mode controller gains. A practical way is suggested to minimise chattering by using lower hard gains emanating from the optimisation of a certain parameter in Huber's function. Hence, a unified framework for system modelling and controller design is established in which controller design requirements have been embedded into the modelling process. Hence, a unified framework for nonlinear controller design is formulated using sliding mode control and neural networks. Using a design example it is illustrated that the control scheme works in practice. The validity of various ways to determine bounds on the model uncertainty arising from neural network training is illustrated through design examples. So far, the modelling part has been performed using FFNN only, which do not possess internal states, as they are static in nature. The next chapter will look into the use of dynamic parameter estimation for sliding mode control.
Chapter 5

Sliding Mode Based Parameter Estimation

5.1 Introduction

Chapter 4 considered the use of neural networks to obtain a GCCF representation for a given system. It should be noticed that the estimation technique is static in itself. It is mainly concerned with the identification of the function \( \eta(.) \) in (3.2). Furthermore, neural network training requires a large amount of plant input output data which may be impracticable from the industrial point of view, and costly in terms of time and computing. In this chapter, a technique for identifying the whole dynamic system is considered. First, a comprehensive survey of the existing relevant literature is presented in Section 5.2. A new sliding mode based robust parameter estimation method is proposed in Section 5.3 and illustrated through a design example in Section 5.4. This estimation scheme is integrated with the robust controller design procedure given in Section 3.4 and its asymptotic stability is proved in Section 5.5. The effectiveness of this method is demonstrated through a design example in Section 5.6. Finally, the chapter is summarised and concluded in Section 5.7.

5.2 Literature review

Nonlinear system identification, unlike linear identification, is a developing field. The problem has been investigated from different viewpoints, each having some advantages and disadvantages. However, no consensus has yet been made as to the best approach.
The problem becomes more severe if the identified nonlinear model is to be restricted to a specific form, such as the GCCF representation. Dynamic neural networks (DNN) or recurrent neural networks present one way to identify a dynamic nonlinear continuous-time model. Dynamic neural networks are the neural network structures which have feedback connections from the outputs to the inputs. This enables the network to possess internal states, a characteristic of dynamic systems. A good introduction to DNN can be found in [29]. Rovithakis and Christodoulou [67, 65, 66] proposed a method for the identification and control of a nonlinear continuous time dynamic system using DNN. They proposed an adaptive update law to update the weights of the identifier DNN and used quadratic stability concepts to prove weight convergence. They also showed that the method is robust against unmodelled fast dynamics using singular perturbation theory. The training method has the same problems as mentioned in the previous section. Also, when the DNN structure is restricted to GCCF form it becomes too limited to capture the dynamics of a given system. Kosmatopoulos and coworkers [39, 40] considered higher order neural networks for the same purpose. Though the proposed scheme gives far more complex models, the structural decisions are completely heuristic, so the rationale for considering a generic higher order recurrent network is lost.

The identification problem has also been considered from the adaptive observer point of view. An adaptive observer is an observer which identifies unknown plant parameters, which are usually assumed to appear linearly, during the course of the state estimation procedure. Hedrick and associates [61, 60, 105] considered a system with linear structure and nonlinear functions with linear parameters. They have used their adaptive observer for the parameter estimation of a vehicle suspension system. Marino and Tomei [52] proposed an adaptive observer for linearly parameterised nonlinear systems and proved the exponential parameter convergence using differential geometric concepts.

Nonlinear identification methods have also been considered in the field of adaptive control. Teel and associates [89] used a nonlinear adaptive observer for system regulation and control. Khalil [38] proposed a scheme to use an adaptive parameter estimation scheme with a high gain observer and then used the estimated parameters for state feedback control.
The use of sliding mode methods for parameter estimation based control has also been reported by Sira-Ramirez [79] and Xu and Hashimoto [103, 104]. Sira-Ramirez considered a linearly parameterised nonlinear system, and proposed a sliding surface based on the estimated parameters, thus showing that the system will attain a sliding mode. Xu and Hashimoto propounded another option. They considered a sliding surface based on the state estimation error and proposed a parameter update which ensures the reachability condition is satisfied. Further comments on this reference can be found in Section 5.4. A new sliding mode based parameter identification method which is robust against model uncertainties is proposed in the next section.

5.3 Sliding mode based robust parameter estimation

This section considers a class of systems which cannot be represented in GCCF representation by traditional modelling methods. However, it is assumed that a proper representation of the system can be obtained through a nonlinear model with linearly varying parameters. The unmodelled dynamics can be considered as uncertainty in the model with known bounds. A little reflection shows that this identification framework encompasses a wide range of nonlinear dynamics (at least locally, if not globally). The model to be estimated is:

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 \\
&\vdots \\
\dot{x}_{n-1} &= x_n \\
\dot{x}_n &= W^T \Gamma(x,t) + pu + \Delta(x,t)
\end{align*}
\] (5.1)

where \(x \in \mathbb{R}^n\), \(W \in \mathbb{R}^n\), \(u \in \mathbb{R}\) is the control input and \(p \in \mathbb{R}\) is the input gain. \(\Delta(x,t)\) is the model uncertainty bounded by a constant \(l > 0\):

\[
\|\Delta(x,t)\| < l
\] (5.2)

here it is assumed that \(\Gamma(x)\) is a known nonlinear function and the state vector \(x\) is accessible. The parameter estimator is given as:

\[
\hat{\zeta}_1 = \zeta_2
\]
\[
\begin{align*}
\dot{\zeta}_2 &= \zeta_3 \\
\vdots \\
\dot{\zeta}_{n-1} &= \zeta_n \\
\dot{\zeta}_n &= \hat{W}^T \Gamma(x) + p u + v(x, t) 
\end{align*}
\] (5.3)

where \( \zeta \in \mathbb{R}^n \) is the state vector of the identifier, \( \hat{W}^T \) contains the estimated parameters, and \( v(.) \in \mathbb{R} \) is the identifier input. The error between the system states and the identifier states is defined as:

\[
e = x - \zeta
\]

The parameter estimation error is expressed as:

\[
\phi = W - \hat{W}
\]

The error system can be written as:

\[
\begin{align*}
\dot{e}_1 &= e_2 \\
\dot{e}_2 &= e_3 \\
\vdots \\
\dot{e}_{n-1} &= e_n \\
\dot{e}_n &= \phi^T \Gamma(x) - v(x, t) + \Delta(x, t)
\end{align*}
\] (5.4)

A sliding surface in the error space is proposed as:

\[
c = \sum_{i=1}^{n} b_i e_i
\]

where \([b_n, b_{n-1}, \ldots, b_1]\) are the coefficients of a Hurwitz polynomial with \( b_n = 1.0 \). \( c \) can be rewritten as:

\[
c = \sum_{i=1}^{n-1} b_i e_i + e_n
\]

implying:

\[
e_n = - \sum_{i=1}^{n-1} b_i e_i + c
\]

From (5.4) it is known that:

\[
\dot{e}_{n-1} = e_n
\]
Chapter 5. Sliding Mode Based Parameter Estimation

which implies:

\[ \dot{e}_{n-1} = -\sum_{i=1}^{n-1} b_i e_i + c \]  
\[ \text{(5.5)} \]

Differentiating \( c \) will yield:

\[ \dot{c} = \sum_{i=1}^{n-1} b_i e_{i+1} + \dot{c}_n \]

implying:

\[ \dot{c} = \sum_{i=1}^{n-2} b_i e_{i+1} + b_{n-1} e_n + \phi^T \Gamma(x) - v(x,t) + \Delta(x,t) \]

Substituting for \( e_n \):

\[ \dot{c} = \sum_{i=1}^{n-2} b_i e_{i+1} + b_{n-1} (-\sum_{i=1}^{n-1} b_i e_i + c) + \phi^T \Gamma(x) - v(x,t) + \Delta(x,t) \]

If \( h \) is defined as:

\[ h = \sum_{i=1}^{n-2} b_i e_{i+1} + b_{n-1} (-\sum_{i=1}^{n-1} b_i e_i + c) \]

then:

\[ \dot{c} = h + \phi^T \Gamma(x) - v(x,t) + \Delta(x,t) \]  
\[ \text{(5.6)} \]

Using equations (5.5) and (5.6) the error system (5.4) can be transformed into:

\[ \begin{align*}
\dot{e}_1 &= e_2 \\
\dot{e}_2 &= e_3 \\
& \vdots \\
\dot{e}_{n-1} &= -\sum_{i=1}^{n-1} b_i e_i + c \\
\dot{c} &= h + \phi^T \Gamma(x) - v(x,t) + \Delta(x,t)
\end{align*} \]
\[ \text{(5.7)} \]

If \( \tilde{e} \) is defined as:

\[ \tilde{e} = [e_1 \ e_2 \ldots \ e_{n-1}]^T \]

then the transformed error system (5.7) can be rewritten as:

\[ \begin{align*}
\dot{\tilde{e}} &= A \tilde{e} + Bc \\
\dot{c} &= h + \phi^T \Gamma(x) - v(x,t) + \Delta(x,t)
\end{align*} \]
\[ \text{(5.8)} \]

where \( A \in \mathbb{R}^{(n-1)\times(n-1)} \) is the companion matrix of \([b_{n-1} \ldots \ b_1] \) and \( B \in \mathbb{R}^{n-1} \) is \([0 \ 0 \ldots \ 1]^T \). \( P \in \mathbb{R}^{(n-1)\times(n-1)} \) is defined as the solution of the Lyapunov equation:

\[ A^T P + PA = -I_{n-1} \]  
\[ \text{(5.9)} \]
Theorem 1 The sliding function $c$ and $\dot{c}$ will decay to zero for the following choice of variables:

$$
\begin{align*}
  v &= h + k_d c + k_d_2 \text{sign}(c) + 2B^T P \dot{c} \\
  \dot{W} &= c \Gamma(x) \\
  k &= 0 \\
  k_d &> l > 0
\end{align*}
$$

(5.10)

Proof

Assuming that the real parameter vector $W$ is constant it can be shown that $\dot{\phi} = -\dot{W}$. Including the parameter estimation dynamics (originating from the parameter update law in the above theorem) in the system (5.8) the overall augmented system can be written as:

$$
\begin{align*}
  \dot{c} &= A\dot{c} + Bc \\
  \dot{c} &= h + \phi^T \Gamma(x) - v(x,t) + \Delta(x,t) \\
  \dot{\phi} &= -c \Gamma(x)
\end{align*}
$$

(5.11)

Consider the candidate Lyapunov function for this system:

$$
V = \dot{\epsilon}^T P \dot{\epsilon} + \frac{1}{2} c^2 + \frac{1}{2} \phi^T \phi
$$

(5.12)

Differentiating $V$ along the trajectories of (5.8):

$$
\dot{V} = \dot{\epsilon}^T P \dot{\epsilon} + \dot{\epsilon}^T P \dot{\epsilon} + \dot{c} c + \phi^T \dot{\phi}
$$

From equation (5.8):

$$
\begin{align*}
  \dot{V} &= (A\dot{c} + Bc)^T P \dot{\epsilon} + \dot{\epsilon}^T P (A\dot{c} + Bc) + c[h + \phi^T \Gamma(x) - v(x,t) + \Delta(x,t)] + \phi^T \dot{\phi} \\
  &= \dot{\epsilon}^T (A^T P + PA) \dot{\epsilon} + 2cB^T P \dot{\epsilon} + c[h + \phi^T \Gamma(x) - v(x,t) + \Delta(x,t)] + \phi^T \dot{\phi}
\end{align*}
$$

(5.13)

Using the definitions from the Lyapunov equation (5.9) and of $v$ from equation (5.10):

$$
\begin{align*}
  \dot{V} &= -\|\dot{\epsilon}\|^2 + 2cB^T P \dot{\epsilon} + c[h + \phi^T \Gamma(x) - h - k_d c - k_d_2 \text{sign}(c) - 2B^T P \dot{\epsilon} + \Delta(x,t)] + \phi^T \dot{\phi} \\
  &= -\|\dot{\epsilon}\|^2 + 2cB^T P \dot{\epsilon} + c[\phi^T \Gamma(x) - k_d c - k_d_2 \text{sign}(c) - 2B^T P \dot{\epsilon} + \Delta(x,t)] + \phi^T \dot{\phi}
\end{align*}
$$
\begin{align*}
\dot{\epsilon} &= -\|\epsilon\|^2 + 2cB^T P \epsilon - 2cB^T P \dot{\epsilon} + c[\phi^T \Gamma(x) - k_{c2} c - k_{d2} \text{sign}(c) + \Delta(x, t)] + \phi^T \dot{\phi} \\
&= -\|\epsilon\|^2 + c[-k_{c2} c - k_{d2} \text{sign}(c) + \Delta(x, t)] + \phi^T (\dot{\phi} + c \Gamma(x)) \\
&= -\|\epsilon\|^2 + c(-k_{c2} c - k_{d2} \text{sign}(c) + \Delta(x, t)) + \phi^T (\dot{W} - \dot{\epsilon} + c \Gamma(x))
\end{align*}

Recognising the fact that \(W\), the actual parameter vector, is assumed to be constant so \(\dot{W} = 0\) and using the definition of \(\dot{W}\) in the above equation:
\begin{align*}
\dot{V} &= -\|\epsilon\|^2 + c[-k_{c2} c - k_{d2} \text{sign}(c) + \Delta(x, t)] \\
\dot{V} &= -\|\epsilon\|^2 - k_{c2} c^2 + c[-k_{d2} \text{sign}(c) + \Delta(x, t)]
\end{align*}

The first two terms on the right hand side will always be negative. Assuming \(c \neq 0\) the sign of the third term can be investigated through the following series of steps:

- Since by assumption
\[k_{d2} > 1 > \|\Delta(x, t)\|\]
the sign of the bracketed term \([-k_{d2} \text{sign}(c) + \Delta(x, t)]\) will always be the same as of \(-k_{d2} \text{sign}(c)\) (being the bigger term).

- Now the sign of the term \(-k_{d2} \text{sign}(c)\) is always opposite to that of \(c\).

- This means that the sign of \((-k_{d2} \text{sign}(c) + \Delta(x, t))\) is always opposite to that of \(c\).

- Which implies that:
\[c(-k_{d2} \text{sign}(c) + \Delta(x, t)) \leq 0\]

Hence:
\begin{align*}
\dot{V} &= -\|\epsilon\|^2 - k_{c2} c^2 + c(-k_{d2} \text{sign}(c) + \Delta(x, t)) \\
\dot{V} &\leq 0
\end{align*}

\(\dot{V} = 0\) only if \(\dot{\epsilon} = 0, c = 0\). When \(c = 0\), \(\dot{V} = -\|\epsilon\|^2 \leq 0\), and \(\dot{V} = 0\) implies \(\dot{\epsilon} = 0\). Since \(\dot{V}\) is negative definite with respect to \(\dot{\epsilon}\) and \(c\), the closed loop system is uniformly asymptotically stable with respect to \((\dot{\epsilon}, c)\) as proved by Lu and Spurgeon in [48]. Thus \((\dot{\epsilon}, c, \phi)\) are bounded. \(c\) will decay to zero followed by the convergence of \(\dot{W}\) (resulting from the update law). Besides, the closed loop system is uniformly Lyapunov stable.
which implies that $\hat{W}$ is bounded. Using similar arguments for parameter convergence in adaptive control as in [84], parameter convergence can be proved if the input is persistently exciting. Thus, the parameters will converge to the actual parameter values if the system input $u$ in equation (5.1) is persistently exciting. If it is assumed that

1. $\text{sign}(0) = 0$
2. $\Delta(x, t) = 0$
3. The input is persistently excitatory so that $\phi = 0$

then it can be seen from the definition of the transformed error system (5.8) and Theorem 1 that the equilibrium point of the system is $\bar{e} = 0, c = 0$. However, if $\Delta(x, t) \neq 0$, then the disturbance will be rejected by the switching term, driving the system back to the origin, provided $\Delta(x, t) < 1$. Hence, in the presence of the uncertainties, the origin is not the equilibrium point of the system rather it is the point of attraction for the system.

### 5.4 Design example

The design example chosen is taken from a paper by Xu and Hashimoto [103]. As mentioned in Section 5.2, Xu and Hashimoto proposed a sliding mode control based parameter estimation procedure for nonlinear systems. The author has found considerable difficulty in implementing the proposed method. It is not trivial to get a suitable reachability condition that will make the system slide in the parameter estimation space. Furthermore, the parameters are assumed to be bounded from above and below, which is quite impractical. In order to reduce chattering, the input signal needs to be filtered. It can be seen from the Section 5.3 that bounds on the parameter space are not needed. The careful choice of continuous and discontinuous sliding gains can help to avoid the use of a filter. The proposed estimation scheme is also robust against a wide class of uncertainties characterised by $\Delta(.)$. Finally, the proposed method is applied on the design example used by Xu and Hashimoto [103] to demonstrate the effectiveness and ease of reachability of the proposed method. The system under consideration is:

$$\dot{x}_1 = x_2$$
\[
\dot{x}_2 = \theta_1 \frac{x_2^2}{1 + \|x_2\|} + \theta_2 x_1 (1 - x_2) + 0.8 x_2 \cos(x_1) u
\]

The actual parameter values are:

\[ \theta_1 = 0.6 \]
\[ \theta_2 = 0.6 \]

The robust sliding mode based estimator is simulated with the following parameters:

\[ b_1 = 10 \]
\[ k_{c2} = 20 \]
\[ k_{d2} = 0 \]

The parameters converge to:

\[ \theta_1 = 0.8 \]
\[ \theta_2 = 0.6 \]

Considering the fact that the parameters are initialised using random values and the state error sliding surface is decaying asymptotically, the estimator is showing good convergence properties (Figure 5.1). To ensure persistency of excitation, a random sequence is chosen as the input.

Having established an effective and practical estimation procedure, the next logical step is to incorporate this estimation scheme into an overall controller design framework. The next section addresses this issue.

### 5.5 Sliding mode based parameter estimation and control

This section integrates the robust parameter estimation method for obtaining the GCCF model for a given system with a robust controller design procedure (direct sliding mode design) discussed in section 3.4. This facilitates controller design for systems with nonexistent GCCF models. The model for the system to be identified is the same as given in (5.1). Also, the parameter estimator has the same structure as before (5.3).

The state estimation error is defined as:

\[ e = x - \zeta \]
The parameter estimation error is given as:

\[ \phi = W - \hat{W} \]

A sliding surface in the state estimation error space is proposed as:

\[ c = \sum_{i=1}^{n} b_i e_i \]

where \([b_n \ b_{n-1} \ldots b_1]\) is a monic Hurwitz polynomial. Another term associated with this sliding surface is:

\[ h = \sum_{i=1}^{n-1} b_i e_{i+1} \]

Similarly the sliding surface in the system state space is formulated as:

\[ s = \sum_{i=1}^{n} a_i x_i \]

where \([a_n \ a_{n-1} \ldots a_1]\) is a monic Hurwitz polynomial. Again an associated term is defined here:

\[ k = \sum_{i=1}^{n-1} a_i e_{i+1} \]

This term will be used for later analysis.

**Theorem 5.2**
Sliding surfaces $s$ and $c$ will asymptotically go to zero for the following choice of variables:

$$v = h + k_2 c + k_{d2} \text{sign}(c)$$
$$u = \frac{1}{p} [-k - \hat{W}\Gamma(x) - k_{c1} s - k_{d1} \text{sign}(s)]$$
$$\dot{\hat{W}} = (c + s)\Gamma(x) \quad (5.14)$$

$$k_{c1} \geq 0$$
$$k_{d1} \geq l$$
$$k_{c2} \geq 0$$
$$k_{d2} \geq l \quad (5.15)$$

**Proof**

Consider the candidate Lyapunov function:

$$V = \frac{1}{2}s^2 + \frac{1}{2}c^2 + \frac{1}{2}\phi^T\phi \quad (5.16)$$

Differentiating $V$ along the trajectories of (5.1) and (5.3):

$$\dot{V} = s\dot{s} + c\dot{c} + \phi^T\dot{\phi}$$

$$= s[k + W^T\Gamma(x) + \Delta(x,t) + pu]$$
$$+ c[h + W^T\Gamma(x) - \hat{W}^T\Gamma(x) + \Delta(x,t) - v(.)] + \phi^T\dot{\phi}$$

$$= s[k + W^T\Gamma(x) - \hat{W}^T\Gamma(x) + \hat{W}^T\Gamma(x) + \Delta(x,t) + pu]$$
$$+ c[h + \phi^T\Gamma(x) + \Delta(x,t) - v(.)] + \phi^T\dot{\phi}$$

$$= s[k + \hat{W}^T\Gamma(x) + \Delta(x,t) + pu] + c[h + \Delta(x,t) - v(.)] + \phi^T[s\Gamma(x) + c\Gamma(x) + \dot{\phi}]$$

Again assuming that the actual parameter $W$ is constant:

$$\dot{V} = s[k + \hat{W}^T\Gamma(x) + \Delta(x,t) + pu] + c[h + \Delta(x,t) - v(.)] + \phi^T[s\Gamma(x) + c\Gamma(x) - \hat{W}] \quad (5.17)$$

Assuming the definitions from (5.15):
\[
\dot{V} = s[\Delta(x,t) - k_{c1}s - k_{d1}\text{sign}(s)] + c[\Delta(x,t) - k_{c2}c - k_{d2}\text{sign}(c)] \\
\leq s[l - k_{c1}s - k_{d1}\text{sign}(s)] + c[l - k_{c2}c - k_{d2}\text{sign}(c)] \\
\leq s[l - k_{d1}\text{sign}(s)] + c[l - k_{d2}\text{sign}(c)] - k_{c1}s^2 - k_{c2}c^2 \\
\leq 0
\]

as every term on the right hand side is negative. In the next section this scheme will be illustrated using a design example.

![Figure 5.2: Tunnel diode model simulated with estimator and controller, outputs and its derivatives](image)

**5.6 Design example**

The combined robust estimation and control method proposed in the last section is used for the estimation of the GCCF model for the tunnel diode circuit introduced in Chapter 4, and subsequently a DSM controller is designed based on this GCCF model. The tunnel diode circuit model is:

\[
\dot{x}_1 = \frac{1}{C}[-g(x_1) + x_2]
\]
Chapter 5. Sliding Mode Based Parameter Estimation

Figure 5.3: Tunnel diode model simulated with estimator and controller

Figure 5.4: Tunnel diode model simulated with estimator and controller with different gains
\[
\begin{align*}
\dot{x}_2 &= \frac{1}{L}[-x_1 - Rx_2 + u] \\
g(x_1) &= \sin(x_1) \\
y &= x_1
\end{align*}
\] (5.19)

The parameters are chosen as:

\[ C = R = L = 1.0 \]

The estimator and controller are designed with the following parameters:

\[
\begin{align*}
kd_1 &= 10 \\
k_d 2 &= 10 \\
k_c 1 &= 0 \\
k_c 2 &= 0 \\
a_1 &= 10 \\
b_1 &= 10
\end{align*}
\]

The simulation results are shown in Figures 5.2 and 5.3. It can be confirmed that both control and estimation objectives are achieved. The parameters have converged to satisfactory estimates and the state error value has decayed. The system regulation is also accomplished. The state error sliding surface initially decays at a faster rate than after 4 seconds. This slowness could be accounted for by the coupling between the estimator and controller systems, which can be reduced by making the estimator dynamics much faster than those of the controller. This is achieved in the next simulation run. The next experiment is performed with:

\[
\begin{align*}
k_d 1 &= 100 \\
k_d 2 &= 10
\end{align*}
\]

The continuous gains are zero. This results in the rapid convergence of the error sliding surface to zero(Figure 5.4)).
5.7 Summary

In this Chapter, a new robust way of modelling nonlinear systems is introduced. The method determines the parameters of a nonlinear system with known structure. The parameter estimation is dynamic in nature. Its stability is proven through Lyapunov stability concepts. Unlike previous methods used for parameter estimation, the technique does not presume any knowledge of the bounds on the system parameters. Neither it requires to impose a filter on the estimator. In addition, the uncertainty structure considered is simple and elegant. In that sense the technique provides an alternative way of addressing the robustness problem in the parameter estimation. If singular perturbation analysis [37] is used to model uncertainty, the resultant scheme is too complex to handle practical issues. Furthermore, in practical problems, it is more realistic to put a bound on the modelling uncertainty than to establish a differential equation representing the unmodelled dynamics. The proposed method is shown to work through an example. This technique produces the GCCf form of a given system, which can be utilised for the sliding mode controller design techniques introduced in Chapter 3. As a next step, the parameter estimation and controller design are considered in the same framework. Their stability is analysed and proven. An additional feature of the parameter estimation scheme is that the scheme is robust against bounded uncertainties. Thus, in addition to robust controller design, robust estimation is also achieved. The validity of the scheme is demonstrated through a design example and is shown to work successfully. Through closed loop simulations the inter-relationship between the parameter estimator gains and controller gains is explored.

So far, Chapters 2, 3, 4, and 5 have proposed and discussed sliding mode controller design methods. It has been claimed that these methods retain their good properties even when tested in the harsh industrial environment. Chapters 6 and 7 provide verification of this claim by presenting the design and testing of these controllers on actual industrial plants.
Chapter 6

High Temperature Multi-burner Furnace Control

6.1 Introduction

Multi burner furnaces are commonly used in industry. In such furnaces, burners are used to maintain desired temperature profiles both in terms of time and space. Traditionally, burner/thermocouple pairs are used for temperature control in a SISO fashion i.e each burner is dedicated to control the temperature of one particular portion of the furnace. In this strategy it is assumed that each burner is isolated from the other temperature points being controlled by the other burners. Due to the strong coupling (due to heat transfer through radiation and convection) between the so called isolated zones the scheme does not always give satisfactory results. Having failed to achieve decoupling, burner saturation becomes a problem in multi-burner furnaces. It would be of interest to achieve desired temperature profiles by manipulating the burners in a unified framework i.e in a MIMO framework. Hence, it is logical to look at multi-burner furnace temperature control as a multivariable control problem. At the same time, like many other industrial units, heating plants are nonlinear in their operation. A feasible approach can be to design controllers based on identified linear models. The designed controllers should be robust enough to handle uncertainties in the linear models arising from the nonlinearities in the actual plant. Sliding Mode Control as described in Chapter 2 is a robust nonlinear technique which has already been used for the temperature control of a single burner furnace [19]. This method is employed for controller design. The knowledge of states is needed for this controller, so a nonlinear observer is also
designed. Among the linear controller design methodologies, $H_{\infty}$ linear robust control has proved to be a successful robust control method. A Two Degree of Freedom $H_{\infty}$ controller is also designed using the loop shaping design procedure [32]. These controllers have been designed and tested using a furnace simulation code. This code is discussed briefly in Section 6.3. The code, modified and developed during the course of this research, is a fully nonlinear simulation code which is capable of simulating a wide variety of furnace configurations varying in geometrical structure, construction material and fuel type. In order to use advanced control techniques a linear model of the furnace is needed. The linear identification of the furnace is described in Section 6.5. Section 6.8 explains the design and simulation of the sliding mode controller-observer pair. An $H_{\infty}$ controller is designed and simulated in Section 6.10 for comparison purposes. Section 6.12 describes the design and simulation of a DSM controller for a high temperature furnace based on the design technique introduced in Chapter 5.

6.2 Multi-burner Furnace Control: Literature Review

In this section a literature review of the use of advanced control techniques for the temperature control of multi-burner furnaces is presented. Various design methods have been used for this purpose.

Use of adaptive control is reported by many authors. Omatu [57] has controlled an electric furnace with three burner inputs and three output temperatures using a Self Tuning Control Scheme. In electric furnaces, the heat transfer within the furnace does not take place through convection, thus limiting the coupling. Gawthrop and associates [26] have used Proportional Integral Derivative (PID) controller auto-tuning to control a laboratory scale tube furnace. Sheppard and coworkers [73] have worked on temperature control of a gas furnace. They have identified a neural model of the furnace and used it to design a Generalised Predictive Control (GPC) scheme. In most of these reported references, a single burner is used to control a single temperature point with in a furnace. Thus, the problem of coupling is not present in these cases. Also in the reported adaptive control techniques, the notion of controller robustness does not exist.

Design of a Linear Quadratic Gaussian (LQG) controller for four burner and four output temperatures is reported in [56]. Expert systems are also being used for furnace control.
Lou and coworkers [45] has devised an expert system for gas carburising furnace control. A rule base for controlling combustion in a double burner boiler can be found in [22]. No nonlinear implementation results are given. Sliding mode control has also been used for gas furnace control for the SISO case[19]. Zohdy and Liu [111] employed a sliding mode control scheme for arc furnace control.

### 6.3 Multi-burner Furnace Simulation Code Development

To design controllers for an industrial multi-burner furnace a simulation code is needed for identification experiments and controller testing. The identified linear model is required for controller design. A Fortran simulation program was provided by British Gas plc. The code was originally written by Dr. M. R. Palmer of British Gas plc. The software was unsuitable for controller testing, as it did not have notions of controller inputs and outputs. Instead of using burner voltages as heat inputs, the burners were simulated using the change in the internal energy. This code was further developed by Dr. C. Edwards of Leicester University so that it could be used for SISO controller design studies. Thus the code was capable of simulating a single burner furnace taking only one burner voltage as input and giving out the temperature of a single furnace wall. The routine has been further developed and modified by the author to make it suitable for MIMO controller design studies. Now the simulation code can incorporate simulation of multi-burner furnaces taking more than one burner voltage as input and returning the desired furnace wall temperatures. In addition, the arrangement for the movement of more than one load in the furnace has also been made. The considerable software changes cannot be explained in detail due to the restriction on the circulation of details of the furnace code. Hence only the changes relating to general furnace simulation procedures are explained.

#### 6.3.1 Furnace Simulation Algorithm

The furnace simulation code is based on the Hottel Zone Method. A full description of this method can be found in [17, 63]. An algorithm for the zone method has been proposed by Saimbai and Tucker [70]. The various steps in the algorithm are described
1. Read input data. The input data consists of the geometry of the furnace, the material properties of the furnace, the burner fuel and the radiation exchange areas [90, 17].

2. The load and wall temperatures are initialised to room temperature (288°K), if the furnace is started from a cold start. If simulation of a preheated furnace is needed then these surface temperatures can be initialised from some other prescribed values.

3. Solve for gas temperatures and heat fluxes in each zone.

4. Calculate flow of combustion gases in each zone.

5. Update temperature profiles in wall, load and hearth in each zone using conduction analysis.

6. Change burner fuel flow rates according to the control algorithm implemented.

7. Check the load status. Decide if a new load should be pushed in or the previous load should be moved out.

6.3.2 Flow Calculation

In the existing code the changes in the fuel flow rate are made by multiplying it by a variable which represents the controller output received from the controller implementation. This control variable is multiplied to all the flow variables. The flow calculations are hence updated without solving flow equations each time the loop is run. This strategy works well in the case of a single zone furnace but it does not hold for a multi-zone furnace. The combustion gas flow equation is solved each time the new fuel input is injected into the furnace.

6.3.3 Load Movement

The original code has the capability of simulating load movement in the furnace as well. But there is provision for the movement of only one load. The load movement scheme
has been changed. Now more than one load can be moved within the furnace. The code has also been changed so that the load should not be pushed into the furnace before the furnace gets properly heated up. Once the temperature of the furnace rises up to a predefined temperature then new loads can be admitted.

6.4 Controller Implementation

The simulation code incorporates a controller routine to implement a single input single output sliding mode controller. This routine has been extended to the multivariable case. Two additional controller routines have also been written. One implements a linear robust control routine. The other one incorporates controllers based on direct and indirect sliding mode control. Furthermore, to facilitate complex mathematical computations the routines to interface the simulation with MATLAB and MATHEMATICA have also been written via C codes.

This section has briefly described how the simulation code was developed. The next section explains how this code has been used for controller design and testing.

6.5 Linear Identification

In order to design a controller using design method described in Chapter 2 a linear model of the furnace is required. This model can be found using standard linear identification techniques [2, 96]. The nonlinear furnace model described in Section 6.3 is used to simulate a multi-burner furnace. This simulation code has been employed as a tool to conduct identification experiments. The geometrical configuration of the furnace under study has also been selected to achieve certain objectives. This point will be explained next.

6.5.1 Furnace Configuration Selection

The specifications of a single burner furnace are available through the British Gas Midlands Research Station. These specifications can be used to determine the configuration of a multi-burner furnace which is representative enough to represent an industrial pro-
cess. It is intended to simulate a long furnace model. The long furnace model is a good example of an industrial heating plant and as the name suggests has length considerably greater than the depth or height. The goal is to control the wall temperatures at the longitudinal ends of the furnace. A load is pushed from one end (Hot End Wall) and made to travel along the length of the furnace until it is taken out from the other end (Cold End Wall). If the opposite end wall temperatures are maintained at different values, the load will go through a varying temperature profile. In a multivariable framework several burner voltages are taken as inputs and the opposite end wall temperatures are good candidates for the outputs to be controlled. The furnace specifications should be realistic enough so that the control objectives can be achieved. As two temperature outputs are planned to be controlled so two burner inputs are selected. An exhaust for flue gases is placed near the cold end wall. Keeping these specifications in mind various furnace configurations have been simulated. Finally a configuration has been selected which is realistic enough and capable of providing the temperature gradient between the two end walls. The proposed furnace model has three times the length of the British Gas test furnace. The British Gas test furnace is a single burner furnace with one temperature point to be controlled. The remaining specifications remain as shown in Figure 6.1

![Figure 6.1: Furnace Geometry under Consideration](image_url)
6.5.2 Operating Point Selection

The linear model for the furnace can only be valid along a certain operating point. This operating point should lie well within the linear range of furnace operation. If the operating point lies outside or on the boundary of the operating region, then it would not be possible to identify a linear model showing high fidelity to the furnace, resulting in large modelling errors and bad controller design. There are strong nonlinearities during furnace operation. First of all there are static nonlinearities. The static gain varies with temperature. The relationship between the burner fuel input and the wall temperatures is nonlinear at low and high temperatures. The temperature tends to saturate beyond a certain value. The fuel flow rate of the burner valve varies with valve input voltage in a nonlinear fashion. The range of input voltage as known from the single burner furnace case is between 0 and 2.5V. The test simulation showed that the relation is fairly linear between 0.25V and 1.25V. The nonlinearity below 0.25V can be explained in terms of fluid flow properties. The saturation effects are observed above 1.75V. Therefore, the optimum value of input voltages for model identification has been selected as 0.75V.

The output temperature has been chosen to lie between 600°C to 750°C. The simulation tests have shown that this temperature range can be achieved using the prescribed input voltage range. The determination of the linear range of operation is investigated using a series of Staircase experiments. Figure 6.2 shows the temperature profiles when the input staircase function varies between 1.25V ± 1.0V. The nonlinearities are apparent. The furnace behaviour is quite linear within 0.75V ± 0.5V as shown in Figure 6.3.

6.5.3 Preliminary Identification Experiments

Before conducting the actual estimation experiment preliminary staircase experiments have been performed to get preliminary data about the nonlinear plant to be identified [108]. Through these experiments the following system parameters can be determined:

- Static Gains
- Time Constants
- Delays
Figure 6.2: Furnace Response at Staircase inputs of $1.25V \pm 1.0V$
Figure 6.3: Furnace Response at Staircase inputs of $0.75V \pm 0.5V$
Chapter 6. High Temperature Multi-burner Furnace Control

- Static Nonlinearity

A brief outline of the preliminary experiments is presented here:

1. **Free-Run Experiment**
   In this experiment the inputs are not activated. The process is allowed to run in open loop mode. The running time should be long enough so that the statistical properties of the outputs do not vary i.e. to construct a stationary process. The data obtained is used to determine statistical parameters of the output disturbances. This test is usually conducted on the actual plant. As the simulation is being performed in a noise free environment, this test is not needed.

2. **Staircase Experiment**
   This test is performed to determine the behaviour of the plant at different operating points. A staircase signal is given to each input channel individually and the output is observed. Each step is applied long enough to let the output settle down. The output can be used to infer static gains and hence static nonlinearities. The input staircase signal is bounded between 0.25V and 1.25V. The static gains are computed at each step as a ratio of settled output temperature to the input voltage. Figure 6.4 shows the variation of static gain along with temperature. These gain relationships exhibit saturation at higher temperatures. This experiment has also been used to predict the linear range of furnace operation as described in the previous section.

3. **Step Experiment**
   This experiment is actually part of the staircase experiment. It has been performed to get a better knowledge of system parameters like delays and time constants. Figure 6.5 shows the furnace behaviour. The furnace is started from ambient temperature (cold start) and a step of 0.25V is applied, which is increased to 1.25V after the temperature has settled down. Figure 6.5 shows that there is no delay in the plant. The rise time and time constants computed from the step responses are shown in Table 6.1. These time parameters are used to choose the frequency for the estimation experiment.
Figure 6.4: Static Nonlinearities in the Furnace Response. The static gain (scaled down by factor of 3) is shown by the solid line and the dotted line shows the corresponding steady state temperature.

<table>
<thead>
<tr>
<th>Temperature(°C)</th>
<th>Rise Time $t_R$ (min)</th>
<th>Time Constant $\tau$ (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hot End Wall</td>
<td>15.13</td>
<td>5.6</td>
</tr>
<tr>
<td>Cold End Wall</td>
<td>15.52</td>
<td>6.17</td>
</tr>
</tbody>
</table>

Table 6.1: Specifications of the Furnace obtained from Step Response
Figure 6.5: Delay Estimation using Furnace Step Response. Solid line depicts temperature (scaled down by 500) and dotted line shows input voltage. The other corresponding input is zero.
4. White Noise Experiment

The inputs are activated by white noise signals in this experiment. System bandwidth and delays are determined from the spectral analysis. Both of these parameters have been already found out in the previous experiment, so this experiment is not carried out.

6.6 Model Estimation

Having obtained certain model characteristics, the next step is to perform an experiment to estimate the actual model. The data collected from this experiment will be used for model estimation. The test input signals for this experiment should have the property of persistent excitation i.e the bandwidth of the signal should cover all the frequencies of interest. The range of frequencies of interest is the bandwidth of the plant. For test signal generation several methods are used including Pseudo Random Binary Sequence and White Noise methods [44, 87]. Here a white noise signal has been used as a test signal as it is difficult to generate a PRBS signal. If the sampling time (i.e time period of the test signal) is $T_s$ and the rise time is given by $t_R$ then from [44]:

$$pNT_s > t_R$$

where $p$ is an integer such that $0 < p \leq 4$. $N$ is an integer dependent on the length $l$ of the test sequence defined as:

$$l = 2^N - 1$$

(6.1)

It is obvious from Table 6.1 that the value of $N$ becomes unpractically large. The effect of the size of $N$ on the PRBS generation is discussed in detail in [87]. The white noise test signal has been generated using standard MATLAB commands. Both input channels of the furnace have been excited by separate white noise signals. If $\tau_{\text{min}}$ is the smallest time constant of the system, then $T_s$ can be determined as:

$$T_s = \frac{\tau_{\text{min}}}{k}$$

(6.2)

where $k$ is an integer with a typical value between 3 to 10. The length of the experiment should be about five to ten times the largest rise time [3]. It is desired to design controllers for the furnace using continuous time techniques. Hence the frequency of
the test signals should be high enough to facilitate continuous time approximation of the
discrete time model obtained. The input output data thus obtained is used for model
estimation. Before estimation it is necessary to treat the data so that it is suitable for
identification.

1. Removal of Outliers

   In industrial systems loose electrical connections and other imperfections result
in unexpected peaks in the measured outputs. These peaks or outliers do not
represent the dynamics of the process and have to be removed. The output val­
ues which are unrepresentative, as known from the preliminary experiments, are
clipped. These output slots are replaced using interpolation. The simulation is
performed in a noise free environment so the removal of outliers is not required in
this case.

2. Trend Correction

   Usually in an industrial process the output tends to drift slowly with time. This
variation is called drift or trend. Drift is quite easy to compensate by a controller
but its presence in the estimation data effects the accuracy of the identified model.
Removal of these effects is called detrending. Detrending can be done in two ways.
The first method is to use a low pass filter to compute the trend. The cut off
frequency is determined from the process knowledge. The other method is to use
a bandpass filter with the bandwidth of the process. Hence the low frequency
trends are clipped off. The best straight line fit from the inputs to the outputs is
determined and removed from the output data.

3. Offset Correction and Scaling

   The identification has been performed at a certain operating point. This operating
point or offset comes into the collected data. To capture the dynamics of the
furnace only, these biases have been removed from the signals. Sometimes the
output signals represent different physical quantities and the signals have to be
scaled so that they have a similar effect on the estimated model parameters. The
output signals in the case of the furnace are the same quantity, i.e temperature.
Hence scaling is not required.
An eight state minimum phase model has been estimated using Matlab commands for model estimation. A continuous approximation of the model in Autoregressive form [96] has been obtained. Figure 6.6 shows the output of the model tested for another set of simulation data. The estimation error becomes smaller when the temperature is near the setpoint values. A perfect linear model for the controller design is not necessary because the control design technique is robust in nature i.e. the controller is robust enough to handle model uncertainties. Hence, the fit is good enough to be used for robust controller design.

![Graph showing the comparison of nonlinear and identified models.](image)

Figure 6.6: The comparison of the nonlinear and identified models.

### 6.7 Analysis of the linear model

The eigenvalues of the identified model are:

\[-0.0177 + 0.0484i\]
The pole zero plot is shown in Figure 6.7. The identified model has very slow dynamics. This point is very important for specifying closed loop performance. The rank of the controllability and observability matrix is full, so the system is fully controllable and observable.

For sliding mode controller design a linear model of the heating plant around a fixed operating point is needed. In this section a linear model has been identified using the data obtained from the nonlinear furnace simulation model (Section 6.3). Now a
nonlinear sliding mode controller will be designed in the next section. For this purpose the concepts introduced in Chapter 2 will be exploited.

6.8 Linear model based sliding mode controller design

The identification of a linear model of the furnace has made it possible to use advanced control methods to design a multivariable controller. There are excessive nonlinearities present such as valve saturation and static nonlinearities in the temperature response i.e the variant temperature characteristics at different operating points. The changes in the fuel properties and measurement noise also contribute to the uncertainties. Variable structure control with a sliding mode is well known for its robustness properties. The sliding mode method provides invariance to parameter uncertainties [109]. Hence variable structure control with sliding mode is a suitable choice for controller design. Besides which a sliding mode controller has been successfully implemented on a single burner furnace [88] and so it is logical to consider extending the method to the multi-burner case. In this section the design of a nonlinear controller and observer scheme is presented. The results of this controller implementation are discussed afterwards.

6.8.1 Furnace Controller Design Issues

The problems encountered during the furnace controller design are outlined briefly in this section.

1. Minimising controller effort
   As described previously the burner valves have saturating behaviour. Beyond a certain value of the input voltage the fuel flow rate ceases to increase with voltage. The burner input voltage which is determined by the controller depends on the integral error states introduced through equation (2.12) in Chapter 2. If the error increases, its integral goes on increasing and the associated input voltage also increases proportionately. However, the actual input to the furnace (fuel flow rate) saturates at its peak value. Consequently the controller effort diverges. The divergence of the controller effort affects the observer convergence, leading
to plant instability. Hence, the valve saturation and the nature of the integrator states lead to a wind-up problem. Additionally the peak input voltage value is 2.5V. From the implementation point of view the input should remain well below this value. Hence the prime issue is to get the desired closed loop performance while keeping the input magnitude to a reasonable limit. One way of achieving this is to heavily penalise the inputs in a linear quadratic design. The specifications thus obtained are used to construct the design matrix $M$ determining the reduced order dynamics as in equation (2.20). If the control inputs are penalised lightly, then the resulting controller will have high gain, thus, more susceptible to integral windup. To avoid this problem, the controller inputs should be penalised heavily and the integral error states should be penalised lightly.

2. Closed loop performance
The demand profile as specified by $\Psi$ in equation (2.10) also affects the controller effort. In order to avoid wind-up the demand specifications should be realistic. A good way is to start with the open loop specifications. The fastest eigenvalue of $\Psi$ should lie within the open loop spectrum. Once the controller starts running successfully with these specifications then the spectrum of $\Psi$ can be shifted further towards the left.

3. Overshoot and Time Response The temperature overshoot is highly undesirable in furnaces. The overshoots cause thermal stresses in the load and the furnace due to high temperature transients. The overshoot means wastage of energy and decrease in the production efficiency due to the increase in load heating time, as the load will take time to cool down after an overshoot. By heavily penalising the inputs in the LQ design the overshoot has been minimised. An obvious related problem can be that of a sluggish response. The plant response may be too slow due to the low magnitude of the inputs. For the furnace problem this has not caused difficulties because speed of response is not the main issue in industrial furnaces.

4. Observer Specifications The observer dynamics are specified through the matrix $A_{22}^o$ (Equation 2.33). Traditionally linear observer specifications are ten times
faster than that of the closed loop characteristics.

5. Minimising Chattering  The chattering of the input voltage can be removed by increasing $\delta_c$ in Equation 2.29. It has been observed that if the observer dynamics are more than fifty times faster than that of the sliding surface then chattering increases. The high frequency of the observed state updates reflect through the input channel. It is thus advisable to keep the observer speed within reasonable limits.

After the controller has been designed, it is implemented and tested on the nonlinear simulation code.

### 6.9 Nonlinear Closed loop Simulation

The controller has been implemented within the nonlinear simulation facility. Two differing trial simulation results are presented. The first simulation has been performed with a moving load in the furnace. This represents a major disturbance. The furnace is divided into two zones. Zone 1 contains the Cold End whilst Zone 2 contains the Hot End. The load enters via Zone 1 and is moved forward. Two constant setpoints are defined for the end wall thermocouples. At start-up, the empty furnace is allowed to reach the desired set-points of $745^\circ C$ and $706^\circ C$ for the Hot and Cold end walls respectively. A load is then pushed into Zone 1 - a preheating zone containing the flue. The load is soaked until it reaches the desired temperature. It is then transferred into Zone 2 and a new load is introduced into Zone 1. The cycle is then repeated. Figure 6.8 shows that the tracking of the chosen reference signals is excellent and the plant is seen to recover well from load disturbances. The second simulation aims to evaluate the controller performance for dynamic setpoint tracking. There are several reasons for using changing setpoint tracking. Dynamic setpoint tracking inherently shows the capability of the controller to cope with varying setpoints. It shows the response time of the controller and gives a measure of the decoupling which the controller can induce between different zone temperatures. It also depicts the controller's capability to overcome uncertainties even when it is operating away from the operating region used for controller design. Figure 6.9 shows the temperature profiles. The plant is following the
Figure 6.8: Simulation for Moving Loads
changing setpoints with high fidelity. The burner input is quite low. In view of the
time scale involved, there are no instantaneous changes in the burner inputs. In the

![Figure 6.9: Output Temperatures for Changing Setpoints](image)

next section, for comparison purposes, a two degree of freedom $H_\infty$ controller will be
introduced, designed and tested.

### 6.10 $H_\infty$ Controller Design

The $H_\infty$ control design method is one of the most successful linear robust control tech­
niques. It has been used in a wide range of applications. A Two Degree of Freedom
design is used here, in which the plant is required to track a reference model. To use
this procedure the required closed loop performance is specified only in terms of the
reference model and frequency weighting function [53]. These weighting functions are
selected so that when they are placed in series with the plant model to form a shaped plant, an appropriate loop shape is obtained. First necessary background is presented and then a brief outline of the design procedure is given. The controller is designed and tested. Finally the nonlinear simulation results are presented and comparisons with the performance of the sliding mode controller developed in Section 6.8 are made.

![Controller Effort for Changing Setpoints](image)

**Figure 6.10: Controller Effort for Changing Setpoints**

### 6.10.1 Two Degree of Freedom Design

The frequency domain representation of the shaped plant $G$ is given by the normalised left coprime factorization:

$$ G = M^{-1}N $$

where $M, N \in H^+_{\infty}$ and are coprime and

$$ MM^* + NN^* = I $$
Figure 6.11 shows the two degree of freedom configuration. $M_o$ is the reference model giving the ideal response of the plant. $\Delta_M, \Delta_N \in H_\infty^+$ is the perturbation in the plant model. The controller matrix $K$ consists of $K = [K_1 \ K_2]$. $K_1$ is the prefilter and $K_2$ is the feedback controller. The problem is formulated such that the $L_\infty$ norm of the transfer function from $(r, \phi)$ to $(u, y, e)$ is minimised in the $H_\infty$ optimisation framework.

The transfer function is given as:

$$
p(I - K_2G)^{-1}K_1 \quad K_2(I - GK_2)^{-1}M^{-1}
\begin{pmatrix}
\rho(I - GK_2)^{-1}G K_1 \\
\rho(I - G K_2)^{-1}G K_1 \\
\rho^2((I - G K_2)^{-1}G K_1 - M_o) \\
\rho(I - G K_2)^{-1}M^{-1}
\end{pmatrix}
$$

(6.3)

The prefilter $K_1$ is determined such that:

$$\| R_{y \beta} - M_o \|_\infty \leq \gamma \rho^{-2}$$

(6.4)

where $R_{y \beta} = (I - G K_2)^{-1}G K_1$ is the closed-loop transfer function mapping $\beta \rightarrow y$ and $\gamma$ is the $L_\infty$-norm achieved for the transfer function given in (6.3).

6.10.2 Loop Shaping Design Procedure

The procedure has been explained in [32]. A brief review is presented here.

1. A simple first order reference model is selected for the plant. This is normally a diagonal matrix. The speed of the reference model should be compatible with the limitations of the plant. Otherwise excessive control effort and poor robustness may result.
2. The open loop frequency characteristics are modified to a desired loop shape by using a frequency domain weighting function. This weighting function $W$ will determine the closed loop specifications.

3. Find the optimal robustness criterion $\gamma_o$ as given in [32]. A higher value normally represent poorer robustness properties. In this case a high $\gamma_o$ may be due to the inconsistency of the closed loop specifications with the robustness requirements.

4. Set the scaling factor $\rho$ with in the range $1 \leq \rho \leq 3$.

5. The optimal value of the robustness measure $\gamma$ is computed. It should be within the range $1.2\gamma_o \leq \gamma \leq 3\gamma_o$.

6. The optimal controller is computed. It is post multiplied with the weight $W$ and the prefilter is rescaled. The degree of the final controller should be $\leq \text{deg}(Plant) + \text{deg}(Reference) + 2\text{deg}(W)$.

6.10.3 Controller Design

Two frequency functions are needed to specify the desired system response. For the furnace problem these are selected as follows.

**Reference Model Selection.**

The reference model specifies the time response of the closed loop. The important part of the open loop dynamic lies within $0.05\text{rad/sec}$. Hence a single pole of $0.05\text{rad/sec}$ is selected for each channel. Deviation from this spectrum results in excessive control effort. The overall reference model transfer function matrix from input to output is given as:

$$M_o = \begin{bmatrix}
0.05 & 0 \\
\frac{0.05}{s+0.05} & 0.05 \\
0 & \frac{0.05}{s+0.05}
\end{bmatrix} \quad (6.5)$$

The gain is selected to make the steady state value unity for a unit step response.
Figure 6.12: Singular Values of Shaped & Unshaped Plant
Weighting Function Selection.

Figure 6.12 shows the open loop frequency response. The open loop dc gain is to be increased to ensure good tracking. The cross over frequency also needs to be decreased to increase the phase margin and hence improve robustness. A weighting function with an approximate integrator in both channels is selected with a gain to align the response at the cross over frequency of 0.1 rad/sec. The shaped plant frequency response is shown in Figure 6.12.

\[ W = \begin{bmatrix} \frac{1}{s + 0.001} & 0 \\ 0 & \frac{1}{s + 0.001} \end{bmatrix} \]  

(6.6)

The \( \gamma \) achieved for this configuration is 5.25. This value is higher than normal. It can be reduced by decreasing the cross over frequency. However this will make the response
sluggish. \( \gamma \) can also be reduced by adding zeros near the cross over frequency. The non-linear simulation results show that the controller is coping well with the uncertainties in the plant. Thus no attempt was made to reduce \( \gamma \) further.

The controller is in a discrete time form. This controller has been implemented on the same furnace simulation code. Again two types of simulations are performed. The cases of moving load and varying temperature profile are considered. Figure 6.14 shows the result for moving load. The nature of the trials is as described above. This controller uses more control effort than the sliding mode controller. In the dynamic setpoint test

![Graphs showing performance of controllers](image)

Figure 6.14: Performance of \( H_\infty \) controller for moving load

there is no appreciable difference in the tracking performance of the two controllers as shown in Figure 6.15. Figure 6.16 shows the controller effort.
6.11 Comparative Analysis

For quantitative comparison of the performance of the two controllers two performance indices are defined. If $N$ is the number of data points:

\[\text{Mean Absolute Controller Effort (MACE)} = \frac{\sum \|u\|}{N}\]

\[\text{Mean Absolute Error (MAE)} = \frac{\sum \|e\|}{N}\]

where $u$ is the control input and $e$ is the tracking error. Table 6.2 compares the performance of the controllers during the moving load simulation.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Channel</th>
<th>$H\infty$</th>
<th>Sliding Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACE</td>
<td>Input 1</td>
<td>0.66</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>Input 2</td>
<td>0.97</td>
<td>0.24</td>
</tr>
<tr>
<td>MAE</td>
<td>Output 1</td>
<td>0.03</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td>Output 2</td>
<td>0.05</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Table 6.2: Comparison of controller performance for Moving Load Simulation

The $H\infty$ controller uses more controller effort and hence provides better tracking for the second input channel. For the varying temperature simulation the comparison is given in Table 6.3. Again the $H\infty$ controller effort is greater than that of the sliding mode controller. The tracking performance is not compared because the method for presenting varying demand setpoints is different for each controller. For the moving load case, the comparison is possible due to constant setpoints. It should be noted that
considering the high scale of temperature, the tracking error is negligibly small in both cases.

![Graph of Hot End Temperature and Demand](image1)

![Graph of Cold End Temperature and Demand](image2)

Figure 6.15: Dynamic setpoint tracking by the $H_\infty$ controller

### 6.12 Direct sliding mode controller design

Having successfully designed and tested the linear model based sliding mode controller, a direct sliding mode controller based on the parameter estimation technique proposed in Chapter 5 is presented.

Since the design technique introduced in Chapter 5 is for SISO models, so a furnace configuration with a single burner was chosen for this case study. The input to the system is a single burner voltage and the output is the hot end wall temperature. The design was performed in two phases i.e parameter estimation phase and controller design phase.
Chapter 6. High Temperature Multi-burner Furnace Control

Figure 6.16: Controller Effort
6.12.1 Parameter estimation

First step consisted of estimating the parameters of the nonlinear model of the furnace. Since the dynamics of the furnace are fairly slow, a three state model was adopted, with states consisting of the output temperature and its first two derivatives. A chirp signal was used as input excitation signal. The squashing function $\Gamma(.)$ (equation 5.1) was selected to be a hyperbolic tangent function, because of its known interpolation properties known from its frequent use in neural networks literature. The initial estimates of the parameters are randomly chosen. Figure 6.17 shows the parameter convergence. The state error sliding surface is characterised by the Hurwitz polynomial $[1 4 4]^T$.

6.12.2 Controller design

The second phase consisted of the controller design. The controller was designed on the basis of the model 5.1) with parameters estimated in the last subsection. The Hurwitz polynomial of the sliding surface was chosen to be $[1 10]$ with a pole at $-10.0 \text{ rad/sec}$ which is a reasonable speed for a furnace. The whole design procedure, including the parameter estimation process is done on the error dynamics of the system with the equilibrium position at $500^\circ C$.

6.12.3 DSM controller simulation

Two results are shown here. In the first simulation (Figure 6.18) the controller is tracking the temperature setpoint of $500^\circ C$ which is the setpoint used for parameter estimation. The second simulation test (Figure 6.19) shows the robustness of the controller which is tracking $450^\circ C$ away from its setpoint of design. The Hurwitz polynomial of the sliding surface was $[1 10]$.

6.12.4 Comparison

For the sake of comparison, a varying setpoint profile (similar to the one used before) was used for DSM controller simulation. The results are shown in figures 6.20 and 6.21.
Figure 6.17: Furnace model parameter estimation
Figure 6.18: DSM controller with output tracking of 500°C
Chapter 6. High Temperature Multi-burner Furnace Control

Figure 6.19: DSM controller with output tracking of 450 $^\circ$C
Figure 6.20: DSM controller with output tracking of a time varying profile
Figure 6.21: DSM controller with output tracking of a time varying profile
Chapter 6. High Temperature Multi-burner Furnace Control

The associated performance indices (as defined in Section 6.11) are:

\[ MAE = 0.1619 \]
\[ MACE = 0.2219 \]

It would be unrealistic to compare these values to the results of the linear model based controller as it was designed for a multiburner furnace. However, the results of a linear model based sliding mode controller for a single burner furnace reported in [17] can be used as benchmark. The reported performance criterion are:

\[ MAE = 0.179 \]
\[ MACE = 1.073 \]

The DSM controller is tracking slightly better but at the cost of higher chattering in control.

6.13 Summary

This Chapter has addressed robust controller design, both linear and nonlinear, for multiburner high temperature industrial heating plants. An existing furnace nonlinear simulation code is modified and developed to make it suitable for simulating multiburner furnaces with MIMO controllers. This code is later on used for linear identification and robust controller design. The linear range of operation of a particular furnace configuration is determined through extensive nonlinear simulations. Then a linear model is identified in this operating range. The identification process includes the pre-processing of the input output data, model estimation experiment design and model validation. All these steps have been described in detail. The open loop characteristics of the linear model obtained are analysed and subsequently used for the sliding mode controller design based on the method introduced in Chapter 2. This controller is passed through two kinds of tests. Dynamic temperature setpoint tracking and temperature regulation with moving load. The first test examines the tracking performance of the controller while the second test assesses the disturbance rejection property of the controller. These tests were simulated in the nonlinear simulation facility, developed before, with the controller in closed loop with the furnace. The sliding mode controller shows good tracking
and disturbance rejection properties. Later on, LSDP method is used to design a two degree of freedom $H_\infty$ controller, which is also tested through closed loop simulations. This controller shows good performance too. The two controllers are compared quantitatively. The comparison reveals that the $H_\infty$ exhibits marginally better tracking at the cost of higher controller effort. Later on, a direct sliding mode controller based on the modelling method of Chapter 5 is designed and simulated. This controller shows good performance but the controller effort shows some chattering. Hence, three practical methods for designing controllers for multi-burner furnaces have been demonstrated and validated. Since the first two methods were MIMO techniques, these can be used for temperature and oxygen concentration regulation simultaneously. The decoupling and tracking properties of the controllers make them good candidates for implementation on a multi-burner furnace.

The effectiveness of the sliding mode methods has been demonstrated on an infinite dimensional and highly coupled problem. The next Chapter addresses the issue of sliding mode controller design for the idle speed control of an automobile engine, which is a finite dimensional system with severe disturbances.
Chapter 7

Idle Speed Control Of An Automotive Engine

The design of a robust nonlinear controller for the idle speed control of an automotive engine is described in this chapter. Sliding mode controller/observer design is considered for the first time in this industrial application. The previously established theoretical results are taken into an industrial design framework. The controller has been realised in dSPACE and successfully implemented on a test rig. Later on, the controller design technique of Chapter 5 is used for controller design.

First, the idle speed control problem is described in detail, then a nonlinear controller design based on the ideas introduced in Chapter 2 is developed. Method of Minimum Entropy is used for the hyperplane design. The tuning method is also discussed. Later on, this tuning process is incorporated in an optimisation framework. A nonlinear observer is used for state estimation and the resulting nonlinear controller-observer pair is implemented on an engine test facility. The results of rig trials are presented and discussed. The robustness of the control strategy is confirmed.

7.1 Introduction

Idle Speed Control (ISC) of an automotive engine is a challenging control problem which is currently faced by the automotive industry. Problems arise because of the extreme nonlinearities present in the process. These nonlinearities stem from the long term variations of engine parameters, speed dependent time delays and the change in surrounding variables such as ambient temperature. In addition to the nonlinearities, the presence
of a variety of disturbances aggravates the problem further. Engagement and disengagement of various accessories like air conditioning, headlights, defrosters, cooling fans etc. cause variable loading on the engine. A disturbance rejection strategy is essential for proper speed control. Hence the controller design problem is multidimensional. While exhibiting good tracking performance, the controller should be robust enough to cope with excessive nonlinearities present. At the same time the controller should be capable of rejecting load disturbances.

The control problem has been approached in the past using a number of different controller design techniques. Puskorius and Feldkamp [58, 59] used Recurrent Neural Networks to design a controller for the ISC problem. They used throttle demand and spark advance as control inputs demonstrating good simulation results. Shim et al [74, 75] looked into the problem of reducing the short term fluctuations occurring in the engine idle speed. These ripples are the result of cylinder imbalances. They derived a nonlinear model for the engine and controlled the fluctuations with a period of a few engine cycles using spark advance as a control input. The control strategy was a combination of error detectors, dead zones and PI controllers. Butts, Sivashankar and Sun [7] employed a combined control strategy consisting of feedforward and feedback control. The feedback controller was an LQR controller used for setpoint tracking. Engine speed was controlled through air bypass valve manipulation. The disturbances acting on the engine were assumed to be either directly measurable or at least their engagement and disengagement is detectable. A feedforward controller was designed to minimise the transfer function from the disturbances to the engine speed. The disturbances were assumed to be 'step' in nature. The feedforward controller was aimed at reducing the peak excursions in the engine speed from the setpoint. Some simulation results were given.

Vachtsevanas and his colleagues [91] used fuzzy logic for idle speed control. They developed a scheme of automatic rule generation to develop a two state dynamic model of the engine. To make a rule base, the variables of interest were manifold pressure, engine speed, throttle angle, spark advance and accessory loads. Williams et al [99] designed an $H_{\infty}$ linear robust controller for idle speed control. They identified linear models for engine speed with spark advance, idle valve and external disturbances as inputs. The
identification and the selection of weighting functions is discussed in detail. Brief sim­ulation results for parameter variations were presented. Vesterholm and Hendricks [93] designed and implemented a sliding mode controller on a 1.275l British Leyland engine. They controlled the engine speed using throttle angle. An IO model derived from a two state mean value engine model (MVEM) was employed to design a sliding mode controller. The sliding surface was the weighted sum of the speed error and its integral. The internal states of the system were not used at all. In addition, only throttle angle was used for speed control, or a SISO Controller was designed. The results presented show a considerable residual steady state error after the application of a 25% load. Here the possibilities of using sliding mode control with a discontinuous observer based on linear state space models will be explored.

The sliding mode controller design method introduced in Chapter 2 will be employed for controller design. The controller assumes full state information, so a nonlinear observer is also designed. A Ford 1.6L Zitec engine is used for the design study. It is a four cylinder, four stroke engine. Two engine variables are available to control the engine speed. The first one is the Idle Speed Control Valve (Air Bypass Valve), referred to as ABV in later sections. The second input is the Spark Advance (SPA). Identified linear models from these two inputs to the output (engine speed) were used for controller design. These models were identified on the test rig and provided by Ford Motor Company. The provided models were in discrete time form and had to be converted to continuous time for the controller design phase. The details of the linear identification methods used can be found in [2, 87].

The utility of ISC and the difficulties incurred in this control problem are elaborated in Section 7.2. Continuous time linear models are required to design the idle speed controller. Section 7.3 addresses the issues regarding the acquisition of such models. From the collection of the acquired models, only one model is selected for the sliding mode controller design. A possible way of selecting such a model is suggested and employed in Section 7.4. The theoretical issues like the stability and robustness of the linear model based sliding mode controller design has been fully explored by Edwards and Spurgeon in [20], as described in Chapter 2. But the practical issues regarding the design of such a controller for a practical problem are discussed now in Section 7.5.
Chapter 7. Idle Speed Control Of An Automotive Engine

This Section presents the design algorithm so that all the design choices are clearly related to the controller performance and robustness, thus making the design process transparent for the user. The algorithm is extended to the hyperplane design in Section 7.6. Two approaches to the hyperplane design, Minimum Entropy and LQ design, are described. Their relationship is also explored. Based on this generic algorithm Section 7.7 presents the engine controller design. A specific software-hardware arrangement dSPACE is used for the controller implementation on the actual rig. Various aspects of the controller realisation in dSPACE can be found in Sections 7.8 and 7.9. A different controller configuration is suggested in Section 7.10. Sections 7.11 and 7.12 are about the testing of the controller, both on the nonlinear simulations and the test rig. Finally, a DSM controller is designed, utilising the concepts introduced in Chapter 5 in Section 7.13. The Chapter is concluded and summarised in Section 7.14.

7.2 Idle Speed Regulation Problem

When an automotive engine is idling and an accessory such as headlights, heater, air conditioner etc is turned on, a certain dip in the engine is observed. If there were no compensation arrangements the engine speed would fall drastically resulting in shut down of the engine. Traditionally, in the automotive industry, this problem of speed regulation is solved by increasing the air flow into the cylinder. This increase in the air flow is different for different loads or accessories. This feedforward controller is accomplished through look-up tables. The formulation of these tables demands extensive time and effort. For the techniques used in this thesis, no feedforward control or a priori knowledge of the disturbances is assumed, potentially saving substantial amounts of time and money.

According to Kao and Moskwa [36] (equation (21)) the torsional dynamics of an internal combustion engine are:

\[ I\dot{\omega} = T_i - T_f - T_{load} \quad (7.1) \]

Where \( I \) is the constant equivalent moment of inertia of the engine, \( T_i \) is the indicated torque or torque produced by the engine, \( T_f \) is the opposing torque due to friction and \( T_{load} \) is the torque due to the load applied on the engine. This load may be due to
accessories when idling or the vehicle load during motion. Assume that the engine is idling. \( T_i \) will be relatively low as a very small amount of torque would be needed to maintain the engine speed at the desired level. When an accessory such as the air conditioning system or a heater is turned on, a load torque is applied on the engine decreasing the right hand side of equation (7.1). On the left hand side, \( I \) is constant so the derivative of the engine speed may become momentarily negative, reducing the engine speed to compensate for the lowered net torque. This effect is not observed when the vehicle is moving since the load torque is being compensated by the driver.

The idle speed is regulated by manipulating two engine variables. In Spark Ignition engines the engine speed is regulated by controlling the amount of air consumed during combustion. This is done by throttling the air mass flow rate going into the cylinder through the inlet manifold. A valve is placed in parallel to the throttle to act as a bypass path for the incoming air (Figure 7.1). The opening and closing of this valve is controlled via a voltage signal. The air intake is controlled using this valve during idling, since the throttle is not used during idling. This valve is called the Air Bypass
Valve (ABV). This is one of the inputs to be used for engine speed regulation. The input voltage is sent to the valve as a Pulse Width Modulated signal (PWM). The designed controller varies the duty cycle of the PWM signal which is limited between 0 and 1.

In an engine cylinder, the air-fuel mixture is ignited through the spark plug just before the piston reaches the top end of the cylinder. The position of the piston with respect to the top end of the cylinder affects the output torque and in turn the engine speed. Thus, the timing of the spark ignition or Spark Advance (SPA) is used to regulate the idle speed. However, this input has only a transient effect. It is worth noting that SPA should not be used during steady state conditions as it generates emissions. The limits on this input are largely dependent on the engine speed, but a rough measure on the limit is between 5 degrees to 25 degrees. The unit of this input is the angle of the crankshaft when the spark ignition takes place.

Obviously the output of the system being considered is the engine speed in rotations per minute or \( \text{rpm} \).

The speed of the engine is sensed two times during each revolution, which means that the sampling time is dependent on speed. The higher the speed, the shorter will be the time constant and vice versa. If \( \omega \) is the engine speed in \( \text{rpm} \) then the sampling time \( T_s \) is:

\[
T_s = \frac{30}{\omega}
\]

(7.2)

Thus the effective sampling time of the implemented controller is variable with speed resulting in time delays which affect the phase of the system. The controller should be robust enough to cope with these time delays.

### 7.3 Obtaining Linear Model of The Engine

While the engine is idling, the engine speed may vary from 600 \( \text{rpm} \) to 1100 \( \text{rpm} \). Similarly the amount of load being applied on the engine via the application of automobile accessories can significantly alter the dynamics of the engine. Hence, the major variables that effect the idling engine dynamics are the engine speed and the applied load. Keeping this in view, a whole range of discrete time models identified over the entire envelope of operating conditions (at different speeds and load conditions) were made
available by Ford motor company. Table 7.1 shows the various speed setpoints used for model identification. Table 7.2 explains the definition of the load numbers which represent the load conditions used for linear model identification. The load number varies from 1 to 4 in such a way that the amount of loading is directly proportional to the size of the load number e.g. 4 is the load number for the heaviest load condition and vice versa.

Figure 7.2 depicts the spots at which the linear models were identified in the two dimensional operating space of the engine speed and the load number.

As mentioned earlier the control inputs selected are ABV and SPA, while the output is the engine speed. Discrete time SISO models from each input to the output are provided. Obviously, these models are not directly suitable for sliding mode controller/observer design. A continuous time MIMO model is required for the controller design purpose. The following procedure was devised to obtain such a model from the information provided.

1. The given discrete time models are in a certain format used in MATLAB called
Figure 7.2: The circles show the combinations of the engine speed and the load numbers used as operating points for the model identification.
"theta" format. This is the format used by the MATLAB Identification Toolbox to store an identified model. A data object in the mentioned format stores the identified discrete time model polynomials, the time delays, the model structure and the sampling time used for identification. A whole family of MATLAB commands exists to convert such a data object into a model of the desired form such as the polynomial, state space or transfer function forms.

2. The estimated models are converted from discrete time to the continuous time form. There are various ways in MATLAB to achieve this end. Command "d2c" converts a discrete time state space model into a continuous time model using a "zero order hold" method. Command "d2cm" provides greater flexibility by giving a choice of methods to be used to represent the discrete elements. This command essentially use the bilinear or Tustin’s approximation. The theory behind this conversion is well established and can be seen in an excellent book on discrete time control systems by Ogata [55]. To use the above mentioned commands, the user has to go through a process of extracting discrete time model and sampling time information from the "theta" object and then use this information in the commands mentioned above. "thd2c" provides a simpler solution. It automatically converts the given discrete time object to its continuous time counterpart. However, this method is recommended only for well conditioned models. Finally, command "th2ss" can be used to obtain state space models.

3. After getting the continuous time SISO models from each input to the output of the engine system, the SISO models are combined together to get MIMO models. This is done as follows:

(a) The direct feedthrough matrices of both SISO models are null. Define the SPA to speed model triplet as \((A_s, B_s, C_s)\) and the ABV to speed model triplet is \((A_a, B_a, C_a)\).

(b) Further define \(x_a \in \mathbb{R}^{n_a}\) and \(x_s \in \mathbb{R}^{n_s}\) to be the state vectors of the ABV and SPA model state spaces respectively. Also \(y_a \in \mathbb{R}\) and \(y_s \in \mathbb{R}\) are the outputs of the ABV and SPA models respectively. If \(u_a \in \mathbb{R}\) is the ABV input and \(u_s \in \mathbb{R}\) is the SPA input, then the input to the combined system
is defined as:

\[ u = \begin{bmatrix} u_a \\ u_s \end{bmatrix} \]

(c) Accumulating the two state vectors in an augmented state vector \( x \in \mathbb{R}^n \):

\[ x = \begin{bmatrix} x_a \\ x_s \end{bmatrix} \]

where \( n = n_a + n_s \).

(d) For the combined state vector the system matrix can be given as:

\[ A = \begin{bmatrix} A_a & 0 \\ 0 & A_s \end{bmatrix} \]

(e) From the definitions of the combined state \( x \) and input \( u \), it follows that:

\[ B = \begin{bmatrix} B_a & 0 \\ 0 & B_s \end{bmatrix} \]

(f) From the principle of superposition, if two linear systems are combined together then their outputs can be added together to form the output of the combined system:

\[ y = y_a + y_s \]

which implies

\[ y = Cx \]

where

\[ C = \begin{bmatrix} C_a & C_s \end{bmatrix} \]

where \( A \in \mathbb{R}^{n \times n} \), \( B \in \mathbb{R}^{n \times 2} \) and \( C \in \mathbb{R}^{1 \times n} \).

(g) The triplet \((A, B, C)\) represents the combined system or MIMO model of the engine. No pole zero cancellation occurs while combining the two systems together which implies that there are no redundant states in the MIMO state space model. The SISO models are of second order and the combined model is a four state model.
Chapter 7. Idle Speed Control Of An Automotive Engine

The MIMO models obtained via this approach are used for subsequent controller design. The sliding mode observer/controller is designed on one of these spot models. The selection of the linear model to be used for the controller design is discussed in the next Section.

7.4 Spot Model Selection

<table>
<thead>
<tr>
<th>LOAD → SPEED (rpm)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>-1.799</td>
<td>-0.8637</td>
<td>-1.188</td>
<td>-1.455</td>
</tr>
<tr>
<td></td>
<td>-2.304</td>
<td>-1.494</td>
<td>-1.305</td>
<td>-2.106</td>
</tr>
<tr>
<td></td>
<td>-7.616</td>
<td>-10.66</td>
<td>-7.926</td>
<td>-6.686</td>
</tr>
<tr>
<td>880</td>
<td>-1.086</td>
<td>-4.213</td>
<td>-3.608+3.102i</td>
<td>-4.743+3.09i</td>
</tr>
<tr>
<td></td>
<td>-2.187</td>
<td>-5.908</td>
<td>-3.608-3.102i</td>
<td>-4.743-3.09i</td>
</tr>
<tr>
<td></td>
<td>-16.27</td>
<td>-7.456</td>
<td>-2.778+1.178i</td>
<td>-2.966+0.92i</td>
</tr>
<tr>
<td></td>
<td>-14.37</td>
<td>-1.5</td>
<td>-2.778-1.178i</td>
<td>-2.966-0.92i</td>
</tr>
<tr>
<td>720</td>
<td>-1.514</td>
<td>-2.541+1.408i</td>
<td>-1.937+2.077i</td>
<td>-1.697+1.695i</td>
</tr>
<tr>
<td></td>
<td>-3.162</td>
<td>-2.541-1.408i</td>
<td>-1.937-2.077i</td>
<td>-1.697-1.695i</td>
</tr>
<tr>
<td></td>
<td>-11.42</td>
<td>-4.105</td>
<td>-2.666</td>
<td>-2.473</td>
</tr>
<tr>
<td>650</td>
<td>-2.413</td>
<td>-3.311+1.291i</td>
<td>-2.539+1.902i</td>
<td>-1.654+2.378i</td>
</tr>
<tr>
<td></td>
<td>-4.341</td>
<td>-3.311-1.291i</td>
<td>-2.539-1.902i</td>
<td>-1.654-2.378i</td>
</tr>
<tr>
<td></td>
<td>-5.025+1.98i</td>
<td>-4.097</td>
<td>-2.59</td>
<td>-2.257</td>
</tr>
<tr>
<td></td>
<td>-5.025-1.98i</td>
<td>-5.105</td>
<td>-9.458</td>
<td>-11.15</td>
</tr>
</tbody>
</table>

Table 7.3: Eigenvalues of the system matrices of the spot models.

It is clear from the previous Section that a set of linear models covering the whole range of engine operation envelope during idling is available. Only one of these models is to be chosen for controller design. Table 7.3 shows the eigenvalues of the system matrices of the individual spot models. The eigenvalue spectrum varies over a wide range. This
Figure 7.3: The eigenvalues of transition matrices of the linear spot models identified over the operating range.

view is graphically reinforced by Figure 7.3 which shows the pole positions as a function of the engine speed and the load applied.

The model selected for the controller design does not represent the dynamics under other operating conditions. This causes uncertainty in the model. To keep this uncertainty or perturbations in the dynamics of the selected model over the whole operational envelope to a minimum, it was decided to use the model whose eigenvalues were closest to the mean eigenvalues of the whole spectrum. Two criteria are used for model selection:

1. The deviation of the eigenvalues from the mean eigenvalue of the spectrum.

2. Condition number of the system matrix.

The selection of a spot model with low condition number will prevent any problems during the numerical procedures of the controller and observer design.

Each spot model, being a 4 state model, has 4 eigenvalues. $\lambda_{\text{mean},k}$ is defined as the mean of the $kth$ eigenvalues of all the spot models. Then a criterion for the eigenvalue deviation from the mean eigenvalues can be given as:

$$\Delta_{ij} = \sum_{k=1}^{4} |\lambda_k(A_{ij}) - \lambda_{\text{mean},k}|$$
Chapter 7. Idle Speed Control Of An Automotive Engine

### Table 7.4: Condition numbers of transition matrices of the linear spot models identified over the operating range.

<table>
<thead>
<tr>
<th>LOAD → SPEED (rpm)↓</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>70.99</td>
<td>237.3</td>
<td>139.9</td>
<td>93.92</td>
</tr>
<tr>
<td>880</td>
<td>95.15</td>
<td>62.05</td>
<td>45.13</td>
<td>62.34</td>
</tr>
<tr>
<td>720</td>
<td>91.94</td>
<td>58.77</td>
<td>54.34</td>
<td>51.63</td>
</tr>
<tr>
<td>650</td>
<td>60.7</td>
<td>42.52</td>
<td>51.78</td>
<td>54.91</td>
</tr>
</tbody>
</table>

### Table 7.5: Pole deviations from mean pole position for all spot models.

<table>
<thead>
<tr>
<th>LOAD → SPEED (rpm)↓</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>8.112</td>
<td>36.09</td>
<td>21.45</td>
<td>7.235</td>
</tr>
<tr>
<td>880</td>
<td>16.09</td>
<td>16.72</td>
<td>16.63</td>
<td>17.55</td>
</tr>
<tr>
<td>720</td>
<td>12.98</td>
<td>7.454</td>
<td>7.237</td>
<td>8.719</td>
</tr>
<tr>
<td>650</td>
<td>11.48</td>
<td>9.813</td>
<td>7.126</td>
<td>7.331</td>
</tr>
</tbody>
</table>

\( i \) and \( j \) are the index numbers of speed and load number respectively. Essentially, \( \Delta_{ij} \) is the sum of the deviations of all the eigenvalues of a given spot model from their corresponding mean position in the envelope spectrum.

The two criteria defined above can be considered for model selection. A model is selected which has minimum \( \kappa(A_{ij}) \) and \( \Delta_{ij} \). Tables 7.4 and 7.5 show the condition number and eigenvalue deviation for each spot model respectively. Figures 7.4 and 7.5 graphically portrays this information.

After comparing the different models it was decided that the spot model relating to the speed of 720 rpm and load number 4 is the one having both \( \kappa(A_{ij}) \) and \( \Delta_{ij} \) close to minimum values. This particular model was thus chosen for the sliding mode controller/observer design.
Chapter 7. Idle Speed Control Of An Automotive Engine

Figure 7.4: The condition numbers of transition matrices of the linear spot models identified over the operating range.

Figure 7.5: Pole deviations from mean pole position for all spot models.
Figure 7.6: The sliding mode observer/controller design algorithm.
7.5 Algorithm for generic sliding mode controller-observer design

This Section presents a design algorithm based on the theoretical results proposed by Edwards and Spurgeon [20]. The theoretical results have already been established but the optimal way to get these results translated into design procedures for practical applications is still an avenue not fully explored. The various parts of the design algorithm are explained in the subsequent sections.

1. The linear model of the system to be controlled can be determined through various means:

(a) To get a linear model of the system using physical principles concerning the system.

(b) If the model obtained is nonlinear then it can be linearised by assuming certain variables in the system to be constant (called setpoint variables). Good examples of this method can be found in textbooks on control engineering; for example, the textbook by Dorf [14].

(c) To determine a linear model using system identification. This method has been explained in detail in Section 6.5.

2. For the application of sliding mode controller-observer design the linear model has to be in the "regular form". It is obvious from the structure of the regular form that only $B$ matrix needs to be transformed to $[0 \ B_2]$, so that the input channels only affect $m$ states, thus isolating the inputs from the remaining $n - m$ states. A numerical technique called QR factorisation is utilised to find a transformation matrix such that $B$ can be factorised into:

$$B = TB_z$$

where:

$$B_z = [0 \ B_2]^T$$

A good treatise on QR factorisation can be found in [27]. The basic idea of QR factorisation is that if a matrix $X \in \mathbb{R}^{j \times k}$ is of full rank, such that $j > k$ then
where $Q \in \mathbb{R}^{j \times j}$ is an orthonormal matrix and $R \in \mathbb{R}^{j \times k}$ is an upper triangular matrix with first $j$ rows to be triangular or non-zero. The QR factorisation is applied to the input distribution matrix $B$ to find a transformation such that $B$ is transformed to $[B_2 \, 0]$, with appropriate flipping of $Q$ and $R$ the $B_z$ can be determined. This transformation matrix is used to transform the given state space model to the regular form. The transformation matrix obtained from the algorithm is orthogonal so it will be invertible thus facilitating state space transformation which needs the inverse of the transformation matrix, since the inverse of the transformation matrix is its transpose, no computation would be needed for finding its inverse. The MATLAB command $qr$ can be used for QR factorisation.

3. This step corresponds to the determination of $\Psi \in \mathbb{R}^{p \times p}$ in equation (2.10), which is fixed as a diagonal matrix with the desired eigenvalues on its diagonal. These eigenvalues correspond to the required dynamic response of the system outputs. The controller is required to minimise the error signal $z_e$ in equation (2.12) which is the integral of the error between the demanded output and the plant output. The demanded output $w(t)$ is given by equation (2.10) which is essentially a first order filter acting on the actual demand signal $W$. This signal $W$ is sent to the controller from the external environment. The poles of this filter are specified by the eigenvalues of $\Psi$. The filter serves two purposes. First of all it smoothens the specified demand signal. Secondly, it controls the rate at which the demand is presented to the controller. If a discontinuous signal such as a step demand is given to the controller, it may produce a large controller effort resulting in input saturation. A modified form of equation (2.12) is:

$$\dot{z}_e(t) = K(w(t) - y(t))$$  \hspace{1cm} (7.3) 

where $K \in \mathbb{R}^{p \times p}$ is a positive definite matrix, acting as a scaling constant on the output error.

4. The output integral error states are augmented at the top of the transformed state space to make the augmented state space as described in equation (2.16).
5. Once the state space system has been transformed and augmented, it can be utilised for hyperplane design. Hyperplane design or to determine the sliding surface matrix \( S \) is a big topic in itself. Due to its length it will be treated in a separate section (Section 7.6). The outcome of this hyperplane design would be \( S \) partitioned into \( S_1 \) and \( S_2 \) used in equation (2.17) such that:

\[
S = [S_1 \ S_2]
\]

6. The hyperplane design is used to get the second transformation so that the sliding surface function \( \eta \) is a part of the transformed state vector. This transformation is defined in equation (2.21):

\[
T_{\eta} = \begin{bmatrix}
I_n & 0 \\
S_1 & S_2
\end{bmatrix}
\]

7. Specify a stable matrix \( \Omega \in \mathbb{R}^{m \times m} \) which specifies the range space dynamics. \( \Omega \) should have eigenvalues considerably faster than the hyperplane eigenvalues. As long as these dynamics are faster than the hyperplane dynamics, they do not effect the performance.

8. Determine \( P \) from the Lyapunov equation:

\[
P^{-1}\Omega^T + \Omega P^{-1} = -Q
\]

for some positive definite matrix \( Q \).

9. The transformed state space matrices are used to obtain the state feedback gains for the linear part of the controller as done in the equation 2.25.

10. Transform the regular form of the system (2.9) through \( T_o \) as suggested in the equation (2.31). This will fix the observer gain \( O \) (equation (2.33)).

11. The design engineer is required to specify two design variables for computing the observer parameters. The first one is \( A_{22}^o \) used in equation (2.33). The specification of this matrix determines the free poles of the observer. Ideally they should be as fast as possible. However, the speed of the observer dynamics is constrained by
the associated increase in computation time and the need to avoid chattering in
the output due to the fast update. The second parameter is $\gamma_0$ used in (2.36). It is
positive scalar constant. Initially it should be set to some small value, and should
be increased during simulation to obtain a satisfactory observer output error.

12. Compute the Lyapunov matrix of $A^o_{22}$ and name it $P_2$.

13. Find $L$ as:

$$L = (P_2C_2B_2)^T$$

14. The nonlinear controller design parameters are $k_{g1}$, $\gamma_c$ and $\Lambda$. $\Lambda$ acts as a scaling
variable, hence it does not have any effect on the dynamics. It can be set to any
appropriate scaling factor. The other variable $k_{g1}$ is a positive scalar constant. It
is constrained by:

$$k_{g1}\kappa(\Lambda) < 1$$

If $\Lambda$ is chosen as an identity matrix of dimension $m \times m$ multiplied by some
scaling factor, then $k_{g1}$ can be chosen between 0 and 1. The other parameter $\gamma_c$ is
another positive scalar constant. This should be fixed to some small value and can
be increased during tuning to get good disturbance rejection properties. It acts
as a nonlinear gain. Care should be taken as selecting a high value may result
in input chattering or saturation. The initial values of $\gamma_c$ and $\gamma_o$ can be safely
assumed to be 0.1. If the output error of the observer does not converge then
increase $\gamma_o$. On the other hand, if the system does not slide then increase $\gamma_c$. $k_{g1}$
should have a positive value smaller than one. It has been observed that if $k_{g1}$ is
set close to 1.0 then it tends to destabilise the closed loop system.

15. $\rho_o(.)$ is given as:

$$\rho_o(U_L, y) = \frac{k_{g1}\|u_L\| + k_{g1}\gamma_c\|\Lambda^{-1}\| + \alpha(t, y) + \gamma_o}{1 - k_{g1}\kappa(S_2B_2)}$$

If bounds on the model uncertainty are not available then $\alpha(.)$ can be taken as
zero.
16. The next step is to design the nonlinear part of the observer which is rewritten here:

\[ v_\nu = \begin{cases} 
-\rho_o(u_L, y) \frac{L C_z e_y}{\|L C_z e_y\|} 
& \text{if } L C_z e_y \neq 0 \\
0 & \text{otherwise} 
\end{cases} \quad (7.5) \]

17. The nonlinear part of the controller is given as:

\[ u_N = \begin{cases} 
-\rho_N(u_L, y) \Lambda^{-1} \frac{P_s}{\|P_s\| + \epsilon} 
& \text{if } s \neq 0 \\
0 & \text{otherwise} 
\end{cases} \quad (7.6) \]

where:

\[ \rho_N(u_L, y) = \|\Lambda\| \rho_o(u_L, y) + \gamma_c \quad (7.7) \]

### 7.6 Hyperplane Design

Once the system is driven into the sliding mode, it should exhibit the properties of the linear system in (2.20) which is specified by \( S \) in equation (2.17). The matrix \( S \) in turn is determined from \( M \). From equation (2.20) it is obvious that the reduced order dynamics (the subsystem pertaining to \( z_1(t) \)) are specified by \( M \), where \( M \) is acting as a state feedback gain matrix. Any state feedback design procedure can be used to determine \( M \). There could be many ways to design the state feedback gain. For a system with states having well defined physical meanings methods like eigenstructure assignment are useful. On the other hand, if the model states do not possess real physical interpretations and no clear performance specifications are given then methods such as Linear Quadratic Regulation (LQR), Linear Quadratic Gaussian (LQG) design or Minimum Entropy (ME) design are useful to design the state feedback gain \( M \) so that the controller shows good performance and sufficient robustness.

It has been shown by Wilkinson on page 87 of his book [98] that for a given matrix \( A \) the sensitivity of its eigenvalues to the perturbation in \( A \) is bounded by the condition number of its eigenvector matrix \( W \), defined as:

\[ \kappa(W) = \|W\| \|W^{-1}\| \]

where \( \|W\| \) is defined as the maximum singular value of \( W \). Hence to make a system robust against the perturbation in its \( A \) matrix it is desirable to minimise the condition
7.6.1 Linear quadratic design

A good treatise on linear quadratic (LQ) design can be found in the textbook on state space analysis by Bernard Friedland [23]. This controller design technique is based on a linear model with the usual system triplet \((A, B, C)\). The objective of the design is to minimise a cost function given as:

\[
J(x_0, u, t_0) = \int_{t_0}^{\infty} x^T Q x + u^T R u \tag{7.8}
\]

where \(x \in \mathbb{R}^n\), \(Q \in \mathbb{R}^{n \times n}\) is a symmetric semi positive definite matrix, \(u \in \mathbb{R}^m\) and \(Q \in \mathbb{R}^{m \times m}\) is a positive definite matrix. It can be seen that by the careful selection of the elements of \(Q\) the magnitude of the states can be controlled. Similarly, the inputs \(u\) can be manipulated by changing the entries in \(R\). The negative state feedback gain for which this cost function is minimum is given as:

\[
K = R^{-1} B^T P
\]

where \(P\) is the solution of the following Riccati equation:

\[
A^T P + PA - PBR^{-1} B^T P + Q = 0
\]

In short, this method essentially consists of weighting system states and inputs, according to the design specification. The design can be accomplished using standard Matlab commands.

7.6.2 Minimum Entropy Design

The Minimum Entropy Design procedure is a variant of LQG controller design [4] which is used to determine a state feedback gain. It is quite similar to the LQ design procedure,
except two Riccati equations are introduced in the solution instead of one. This method consists of the optimisation of two cost functions, one for controller design and the other for observer design. For a system triplet \((A, B, C)\) with their usual dimensions, the algorithm for selecting the feedback gain is described below:

1. Choose appropriate weighting matrices \(Q \in \mathbb{R}^{n \times n}\) and \(W \in \mathbb{R}^{n \times n}\) for the system states. \(R \in \mathbb{R}^{m \times m}\) is the weighting matrix for the inputs and \(V \in \mathbb{R}^{p \times p}\) is the output weighting matrix. The choice of these matrices will be explained in the next section.

2. Select the value of a positive scalar \(\gamma\). This constant is a measure of the robustness in the system. A lower value of \(\gamma\) means higher robustness.

3. Using the Matlab Robust Control Toolbox command 'aresolv', solve these two Riccati equations:

\[
A^T X + X A - X (BR^{-1}B^T - \gamma^{-2}W )X + Q = 0
\]
\[
AY + Y A - Y (C^T V^{-1}C - \gamma^{-2}2Q)Y + W = 0
\]  

(7.9)

to obtain \(X\) and \(Y\).

4. If the solutions to these Riccati equations exist, then reduce the value of \(\gamma\) and solve the equations again. In this way, find the least \(\gamma\) for which the solutions exist.

5. The state feedback gain matrix \(K_{sfb}\) is given by:

\[
K_{sfb} = R^{-1}B^T X (1 - \gamma^{-2}YX)^{-1}
\]  

(7.10)

Here, it should be emphasised that this design procedure also computes an associated observer gain. As the observer being used here is a sliding mode observer, this observer gain is not relevant within the sliding mode design procedure.
7.6.3 Relationship between LQ and Minimum Entropy Cost Functions

As mentioned earlier, the cost function optimised during LQ controller design is given as:

\[
J(x_o, u, t_o) = \int_{t_o}^{\infty} x^T Q x + u^T R u
\]  \hspace{1cm} (7.11)

In contrast, the cost function minimised by the minimum entropy controller design is the \(\gamma\) entropy of the system which is defined by [4] :

\[
I_\gamma(H) = \begin{cases} 
\frac{1}{2\pi} \int_{-\infty}^{\infty} \sum_{i=1}^{n} - \gamma^2 \log(1 - (\sigma_i(H(j\omega))/\gamma)^2) d\omega & \text{if } \|H\|_{\infty} < \gamma \\
\infty & \text{if } \|H\|_{\infty} \geq \gamma
\end{cases}
\]  \hspace{1cm} (7.12)

here \(H(j\omega)\) is the transfer function of the system under consideration, \(n\) is the order of the system and \(\sigma_i\) is the \(i\)th singular value of the system. This norm is a "maximum" when it is bigger than \(\gamma\), so any sensible optimisation approach applied on this norm will try to maintain \(I_\gamma(H)\) always less than \(\gamma\). \(\gamma\) is a positive number specified by the designer and so provides freedom for the designer to select the value of \(\gamma\) to achieve criterion other than cost minimisation.

Consider the closed loop system configuration shown in Figure 7.7 with system \(G\) and the controller \(K\). \(w\) is the external input or disturbance to the system, \(u\) is the control input, \(y\) is the system output to be feedback and \(z\) is another output used to monitor the performance of the controller. \(w\) is assumed to be a noise input. A sensible criterion for the minimisation of the cost function would be to reduce the effect of \(w\) on the cost.
function in equation 7.8 to minimum. Boyd showed in his book [4] that if the output
signal \( z \) is defined as:

\[
z = \begin{bmatrix} R^{1/2} u \\ Q^{1/2} x \end{bmatrix}
\]

(7.13)

Then the minimisation of the cost function in (7.8) is equivalent to the minimisation of
the \( H_2 \) norm of the transfer function from \( w \) to \( z \), so the cost function is equivalent to:

\[
J(.) = \| H \|_2^2
\]

where \( H(.) \) is the mentioned transfer function. In both LQ and minimum entropy (ME)
design the norms of the transfer function from \( w \) to \( z \) are minimised. The \( H_2 \) norm is
minimised during LQ controller design while minimum entropy norm \( I(.) \) optimisation
takes place in ME design.

D. Mustafa [54] claimed that both design methods can be considered in the same frame­
work (similar to the one shown above) and proved that:

\[
J(.) \leq I(.)
\]

which is quite significant as \( I(.) \) would serve as the upper bound on the LQ cost function.

In terms of singular values the \( H_2 \) norm is:

\[
H_2 = \left( \frac{1}{2\pi} \int_{-\infty}^{\infty} \sum_{i=1}^{n} \sigma_i(H(j\omega))^2 d\omega \right)^{1/2}
\]

(7.14)

Recalling the inequality:

\[
-log(1 - x^2) > x^2
\]

This inequality is true for \( \|x\| < 1 \). Defining:

\[
x = \left( \frac{\sigma_i}{\gamma} \right)
\]

implies that:

\[
-log(1 - (\frac{\sigma_i}{\gamma})^2) > (\frac{\sigma_i}{\gamma})^2
\]

for \( \|(\frac{\sigma_i}{\gamma})\| < 1 \). This relationship is true for all singular values less than \( \gamma \). If any singular
value is greater than \( \gamma \) then \( I(.) \) is \( \infty \). Exploiting equations (7.8), (7.12), (7.14) and the
above mentioned identity:

\[
I_{\gamma(H)} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \sum_{i=1}^{n} - \gamma^2 log(1 - (\sigma_i(H(j\omega))/\gamma)^2) d\omega
\]
Chapter 7. Idle Speed Control Of An Automotive Engine

Using the identity:

\[ I_{\gamma(H)} > \frac{1}{2\pi} \int_{-\infty}^{\infty} \sum_{i=1}^{n} \gamma^2 \left( \frac{\sigma_i(H(j\omega))}{\gamma} \right)^2 d\omega \]

Cancelling \( \gamma \) terms:

\[ I_{\gamma(H)} > \frac{1}{2\pi} \int_{-\infty}^{\infty} \sum_{i=1}^{n} (\sigma_i(H(j\omega)))^2 d\omega \]

from the definition of the \( H_2 \) norm:

\[ I_{\gamma(H)} > J(.) \]

Thus, the minimum entropy norm overbounds the quadratic norm. Hence, by ME norm optimisation quadratic cost function minimisation can be achieved.

The optimisation of the transfer function from the input disturbance to the output performance exhibits the robustness of the controller. Since \( w \) is assumed to be absent in the standard LQ configuration, this configuration (commonly called Linear Quadratic Regulation or LQR) does not address the issue of robustness. In the Linear Quadratic Gaussian method the disturbance is considered as white noise; the quadratic norm of the disturbance transfer function is minimised with the tunable parameters affecting only the performance of the system which implies that only the performance of the system can be improved with the controller tuning so it is not possible to achieve the desired robustness with LQG control. In the ME design method, both elements i.e performance and robustness are considered. Required performance is obtained by the choice of proper weightings on the system states and the inputs. At the same time \( \gamma \) can be tuned to achieve the desired robustness. Hence the two tunings are employed together to arrive at a satisfactory compromise between the robustness and the performance.

7.6.4 Determining the hyperplane from the resulting state feedback gain

The state feedback gain matrix defining the switching function can be determined in two ways. Either it is designed for the reduced order system, i.e the \( (A_{11}, A_{12}) \) pair in equation (2.15) or for the whole system. For the reduced order system, the computation of \( S \) is quite straightforward. \( S \) is partitioned into \( S_1 \) and \( S_2 \) as in equation (2.17). \( S_2 \) can then be found from equation (2.26). Assign the state feedback gain \( K_{sfb} \) to be \( M \).
Now $S_1$ can be computed from equation (2.19).

Some modification is required when the state feedback gain relates to the whole system. The computation procedure can be decomposed into these steps:

1. The linear closed loop system is represented by:

   \[ A_{cl} = \hat{A} + \hat{B}K_{sf} \]

   where $\hat{A} \in \mathbb{R}^{n+p \times n+p}$ is the system matrix of the augmented system defined in the equation (2.31), and

   \[ \hat{B} = \begin{bmatrix} 0 \\ B_2 \end{bmatrix} \]  

   (7.15)

   where $B_2$ and $\hat{T}$ have been defined in equation (2.15). $K_{sf}$ is the state feedback matrix for the augmented system. Define $n_1 = n + p$ to be the new system dimension.

2. Choose the $n_1 - m$ closed loop eigenvalues (of $A_{cl}$) which are to be used for hyperplane design. Usually the faster set of $m$ eigenvalues is kept for the range space dynamics, so the remaining $n_1 - m$ eigenvalues will specify the reduced order system dynamics.

3. Let $R_e$ be the matrix containing the eigenvectors corresponding to the $n_1 - m$ eigenvalues selected above. The eigenvalue and eigenvector computations can be easily done using the Matlab command `eig`.

4. Perform a singular value decomposition on the matrix $R_e$ such that:

   \[ R_e = U_r \Lambda_r V_r \]

   where $\Lambda_r$ is a diagonal matrix with the singular values of $R_e$ on its diagonal, $U_r$ and $V_r$ are unitary matrices. The Matlab command `svd` provides this decomposition.

5. Since the chosen $n_1 - m$ eigenvectors should lie in the null space of $S$, it follows that:

   \[ SR_e = 0 \]
or

\[ SU_r \Lambda_r V_r = 0 \]

Since \( V_r \) is orthogonal:

\[ SU_r \Lambda_r = 0 \]

Partitioning \( \Lambda_r \) and \( U_r \), such that:

\[ \Lambda_r = \begin{bmatrix} D \\ 0 \end{bmatrix} \] (7.16)

\[ U_r = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \] (7.17)

where \( D \in \mathbb{R}^{(n-m) \times (n-m)} \), \( U_1 \in \mathbb{R}^{n \times (n-m)} \) and \( U_2 \in \mathbb{R}^{n \times m} \). The null part in \( \Lambda_r \) allows us to drop \( U_2 \).

6. It is easy to see:

\[ SU_1 D = 0 \]

since \( D \) is of full rank:

\[ SU_1 = 0 \]

as \( U_1 \) is unitary:

\[ U_2^T U_1 = 0 \]

It implies that:

\[ S = \begin{bmatrix} 0 & I_m \end{bmatrix} U_r^T \] (7.18)

will make:

\[ SU_1 = 0 \]

The next Section describes design of a sliding mode controller-observer pair for the ISC problem.
Chapter 7. Idle Speed Control Of An Automotive Engine

7.7 Engine controller design

The hyperplane design constitutes the most important part in the design as far as the performance is concerned. The choice of the other observer and controller variables affects the running of the controller considerably. These designs will be done in this Section. First, the nonlinear simulation model of the engine is explained, as this model is extensively used for the initial testing of the controller designs.

7.7.1 Nonlinear Model Simulation

The designed controllers are initially tested by simulating the controllers in closed loop with a nonlinear model of the engine. As mentioned earlier, the discrete time linear models of the engine scheduled (with respect to load and speed) were provided. They were linearly interpolated against the setpoint variables so that the model could represent the engine dynamics over a wide envelope. The given nonlinear model is in a discrete time form with no notion of time. The model is driven by the triggering of pulses coming from the engine management system, twice per revolution. In a simulation environment this would mean a unit time step is equal to one pulse. This could be a problem when running continuous time controllers needing an absolute notion of time. This is required also for observer integration. The discrete time model is thus modified to run with a time clock instead of triggered pulses, so that it can simulate a closed loop system in continuous time.

The simulation model is set up such that any desired load can be applied to test the disturbance rejection properties of the controller being simulated. When a load step is applied the engine experiences a dip in speed, which is eventually regulated by the controller. There is a flare in the engine speed when the load step is withdrawn.

The inputs to the engine model are also constrained realistically. ABV varies between 0 to 1 and SPA has a range between 5 degrees and 25 degrees.

7.7.2 Engine Hyperplane Design

To design the hyperplane of the closed loop system with sliding mode controller two methods are used; the Minimum Entropy (ME) design and Linear Quadratic (LQ)
Minimum Entropy based hyperplane design

The ME design is initiated by selecting \( Q, R, W, V \) to be identity matrices of the appropriate dimensions (as defined in the last Section). The first entry of the diagonal of \( Q \) is scaled so that it is significantly bigger than the other entries. This will ensure that the first state, which is the integral of the output error, is penalised. The disturbance rejection property of the controller mainly relies on this weighting. The load disturbance acts on the output. The first impact of the load disturbance is to increase the error state drastically. Thus the tuning of this element of \( Q \), named \( q_{11} \), is very crucial. Since \( R \) represents the weightings on the input, the first entry relates to the ABV input and the second entry on the diagonal of \( R \) weights the SPA input. SPA has very strict limits on its magnitude, so its weighting is kept to unity while the ABV, having less restrictive limits, is weighted by 0.01. It is found through simulations that these values of the elements of \( R \) are a good compromise. The first objective is to avoid input saturation. The inputs should not hit their limits, as it may cause excessive chattering and oscillations. Since the controller output is truncated before going into the engine hardware, the controller would not be able to keep the system on the sliding surface which would deteriorate the performance. The second objective is to keep the weightings on the inputs low enough to enable the controller to recover from the dip in the engine speed after the load disturbance has been applied, thus giving room for sufficient increase in the speed of recovery by tuning \( Q \). \( W \) and \( V \) are the design matrices related to the observer design, but the designed observer is not used for the state estimation, rather a sliding mode observer is designed separately because of its proven closed loop stability with the sliding mode controller [20]. So \( V \) and \( W \) are not tuned. A good starting point for the robustness margin \( \gamma \) is found to be 10 and then it is reduced using the bisection algorithm. It is seen that for this particular problem, the solution of the Riccati Equations (See last Section for details) exist for \( \gamma \) as low as 0.7. The scaling gain \( K \) is the weighting on the integral error state defined in (2.12). Its value is chosen to be 1.0 here. By increasing this gain, the controller dynamics can be made faster. The
Chapter 7. Idle Speed Control Of An Automotive Engine

design matrices chosen are:

\[
Q = \begin{bmatrix}
1e0 & 0 & 0 & 0 & 0 \\
0 & 1e -4 & 0 & 0 & 0 \\
0 & 0 & 1e -4 & 0 & 0 \\
0 & 0 & 0 & 1e -3 & 0 \\
0 & 0 & 0 & 0 & 1e -4
\end{bmatrix}
\]

\[
W = \begin{bmatrix}
1e -2 & 0 & 0 & 0 & 0 \\
0 & 1e -2 & 0 & 0 & 0 \\
0 & 0 & 1e -2 & 0 & 0 \\
0 & 0 & 0 & 1e -2 & 0 \\
0 & 0 & 0 & 0 & 1e -2
\end{bmatrix}
\]

\[
R = \begin{bmatrix}
1e -2 & 0 \\
0.0 & 1e0
\end{bmatrix}
\]

\[
V = \begin{bmatrix}
1e -1 & 0 \\
0 & 1e -1
\end{bmatrix}
\]

As discussed before, the tuning of the first element of \(Q (q_{11})\) greatly effects the controller disturbance rejection property. If \(q_{11}\) is too small then the speed recovery after the load application would be slower. On the other hand, very large \(q_{11}\) causes input saturations and oscillations. Keeping these two points in mind an extensive sweep on \(q_{11}\) is performed to find a suitable \(q_{11}\). Table 7.6 shows the condition number of the eigenvector matrix of the closed loop system matrix obtained from the controller design performed across the range of \(q_{11}\). These designs are then tested on the nonlinear simulation model. During the simulation, the maximum possible load is applied and the engine speed response is monitored. The first column in the Table 7.6 is the mean of the absolute output error observed. From the Table it is obvious that the controller will show low output error when \(q_{11}\) is lower than 19 and the condition number is below 200. The comparative engine speed response of three controllers showing good performance is shown in the Figure 7.8. The controller corresponding to \(q_{11} = 14\) has good speed recovery without any overshoot. From the Table 7.6 it is seen that the controller with \(q_{11} = 14\) has the lowest condition number and mean output error. The design with
<table>
<thead>
<tr>
<th>$q_{11}$</th>
<th>Mean Absolute Output Error</th>
<th>Cond. No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.4</td>
<td>136.4</td>
</tr>
<tr>
<td>4</td>
<td>552.8</td>
<td>270.9</td>
</tr>
<tr>
<td>5</td>
<td>11.68</td>
<td>129.1</td>
</tr>
<tr>
<td>6</td>
<td>11.74</td>
<td>125.8</td>
</tr>
<tr>
<td>8</td>
<td>767.1</td>
<td>274.5</td>
</tr>
<tr>
<td>9</td>
<td>11.63</td>
<td>138.3</td>
</tr>
<tr>
<td>10</td>
<td>11.92</td>
<td>112.6</td>
</tr>
<tr>
<td>11</td>
<td>19.96</td>
<td>52.44</td>
</tr>
<tr>
<td>12</td>
<td>13.57</td>
<td>88.22</td>
</tr>
<tr>
<td>13</td>
<td>355.8</td>
<td>214.5</td>
</tr>
<tr>
<td>14</td>
<td>12.68</td>
<td>94.16</td>
</tr>
<tr>
<td>15</td>
<td>435.5</td>
<td>253.5</td>
</tr>
<tr>
<td>16</td>
<td>293</td>
<td>284.2</td>
</tr>
<tr>
<td>17</td>
<td>15.69</td>
<td>149.6</td>
</tr>
<tr>
<td>18</td>
<td>13.15</td>
<td>142.7</td>
</tr>
<tr>
<td>19</td>
<td>296.8</td>
<td>287.3</td>
</tr>
<tr>
<td>21</td>
<td>968.6</td>
<td>80.87</td>
</tr>
<tr>
<td>23</td>
<td>968.6</td>
<td>138.8</td>
</tr>
<tr>
<td>25</td>
<td>816.5</td>
<td>214.3</td>
</tr>
</tbody>
</table>

Table 7.6: The condition number and the output error for different $q_{11}$ in an ME based sliding mode controller.
$q_{11} = 14$ is selected. Figure 7.9 shows a typical nonlinear simulation test. Initially the controller is allowed to run for a certain time so that it drives the engine speed to 880 rpm and reaches steady state. Then after approximately 3 seconds the load disturbance (EEC-LOAD) is applied, as depicted in the figure. The controller drives the speed back to the setpoint in approximately 8 seconds. The load applied represents the heaviest load during idling. It can be seen that the SPA input is not hitting the upper limit (shown by the dotted line), rather it is hitting the lower limit during the steady state, which means the spark offset provided by the controller is negligible. To ensure low emissions, the spark offset should be close to zero during the steady state condition. Figure 7.9 shows the performance of this controller including the controller output and the load applied.

![Effect of weight Q on ME controller](image)

Figure 7.8: Nonlinear simulation results for various ME-based sliding mode controller

**LQ based hyperplane design**

For LQ design $Q$ is taken to be an identity matrix and $R$ is also initialised as an identity matrix. The weighting on the ABV input is reduced to 0.01 so that it can be used with
Figure 7.9: Nonlinear simulation results for successful ME-based sliding mode controller
Table 7.7: The effect of Q weighting on LQ based controller performance.

<table>
<thead>
<tr>
<th>$q_{ll}$</th>
<th>Output Error</th>
<th>Cond. No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>72.64</td>
<td>139.6</td>
</tr>
<tr>
<td>3000</td>
<td>40.15</td>
<td>1274</td>
</tr>
<tr>
<td>5000</td>
<td>26.03</td>
<td>616</td>
</tr>
<tr>
<td>7000</td>
<td>15.54</td>
<td>515.2</td>
</tr>
<tr>
<td>9000</td>
<td>14.32</td>
<td>529.3</td>
</tr>
<tr>
<td>$1.1e+04$</td>
<td>15.78</td>
<td>176.3</td>
</tr>
<tr>
<td>$1.5e+04$</td>
<td>20.13</td>
<td>576</td>
</tr>
</tbody>
</table>

full authority. As in ME design, the first element of $Q$, $q_{ll}$, puts a weight on the state representing the integral of the output error and so this state should be penalised heavily.

Table 7.7 shows the performance of the sliding mode controller with the hyperplane designed by LQ design. Output error is depicted by taking the mean of the absolute error. Lower values correspond to disturbance rejection. At the same time the condition number of the eigenvector matrix of the closed loop system is also shown, exhibiting the robustness of the controller. As explained before, it should be as low as possible.

The table reveals that increasing $q_{ll}$ greatly reduces the output error. However, the condition numbers remain unacceptably large. Such large condition numbers indicate poor robustness.

Figure 7.10 compares the performances of the controllers with different values of $q_{ll}$. The controller simulations with three $q_{ll}$ weightings are shown. The controller with $q_{ll} = 7e3$ is suffering a very small dip when the disturbance is applied and its recovery rate is also better than its counterparts. Figure 7.11 shows an LQ based sliding mode controller simulation. The controller drives the speed back to the setpoint in 10 seconds, which is greater than the time taken by the ME based controller. It can be seen that the SPA input is saturated to its upper limit, which makes this controller unsuitable for the rig implementation; there will be additional uncertainties on the rig, but SPA does not have any room to increase the gain to make the controller more robust. This problem is not encountered in the ME based controller.
Comparatively, the ME based sliding mode controller is considered better than its LQ counterpart due to the following considerations:

1. Looking at the Tables 7.6 and 7.7 reveals that the condition numbers of the LQ based designs is unacceptably large as compared to that of the ME based designs. As explained before, a lower condition number implies that the closed loop eigenvalues are less sensitive to the system perturbations. The ME based design is more robust to the system perturbations and is preferable. It can be argued that the condition numbers in the LQ based designs are large because of the selection of a large value $q_{11}$. However, to ensure fast recovery from the speed dips, these higher values are essential.

2. The second cause of concern is the input saturation in the LQ based controller (Figure 7.11) which would deteriorate the controller performance considerably. The ME based design does not suffer from this problem giving negligible SPA input (Figure 7.9), which ensures low emissions.
Figure 7.11: Nonlinear simulation results for LQ-based sliding mode controller
3. Figure 7.12 shows the speed responses of the best of the two classes of the controllers. Though the ME based controller has a greater dip, it settles down earlier than its LQ counterpart.

![Figure 7.12: Comparison of the LQ and ME based sliding mode controllers](image)

To design an ME based sliding mode controller, a sweep on the first state weight has been reported. One of the purpose of this tuning has been to find the controller with a low condition number. It is logical to design the ME based controller in an optimisation framework by minimising the condition number of the eigenvector matrix of the closed loop system matrix against the weighting matrices. The optimisation gives these weights for the condition number of 15.14.

$$Q = \begin{bmatrix}
1.661e+04 & -5.153e+04 & -7.515e+04 & 9506 & -2.7e+04 \\
2752 & 3536 & -7214 & 1.552e+04 & 2.882e+04 \\
-9317 & 1040 & -680.6 & 1106 & -2.873e+04 \\
2.91e+04 & -8711 & -4818 & 876.9 & 8137 \\
2.964e+04 & -9056 & -7411 & 344.2 & 5562
\end{bmatrix}$$

$$R = \begin{bmatrix}
-1.645e+04 & 2.023e+04 \\
2841 & -576.7
\end{bmatrix}$$
Chapter 7. Idle Speed Control Of An Automotive Engine

The condition number is much better than the previous ME based design. However the controller gives a very poor and sluggish response on the nonlinear simulation test. The exercise reveals that the controller with the lowest condition number may not be the one providing the best performance. In other words, the set of closed loop poles which are minimally sensitive to perturbations may be inappropriate from the point of view of performance.

### 7.7.3 Other Controller/Observer Variables

The observer eigenvalue is set to $-20.0$ and $\gamma_o$ (defined in equation (2.36)) is selected to be $0.5$. These values ensure that the observer gain is fast enough to give a reasonable output error decay. In order to remove chattering, $\delta_o$ from equation (2.36) is taken to be $1.0$.

The engine inputs ABV and SPA used for speed control are severely restricted by their magnitude limits. In addition to this, the engine process is a discrete time process with variable time interval, so it is more prone to input saturation if the controller dynamics are fast. Hence the choice of $\Psi$ (equation 2.10) is made realistically to be $-1.0$, so as not to saturate the inputs.

The linear part of the observer is designed by specifying the matrix $A_{22}$ in equation (2.33). This observer specification should be as fast as possible. However, as the sample rate has to be the same as that used for the controller and the controller sampling rate is fixed. For a fixed sampling rate it is not possible to integrate differential equations with arbitrarily fast dynamics due to integration errors caused by the insufficiently small step size. Hence, there is a limit to which the speed of the observer can be increased. Keeping these considerations in mind, the observer eigenvalue is set to $-20.0$ and $\gamma_o$ (defined in equation (2.36) is selected to be $0.5$. The magnitude of $\gamma_o$ provides robustness to the
state estimation. These values ensure that the observer gain is fast enough to give a reasonable output error decay.

The nonlinear parts of the controller and observer can be fully determined by specifying the scalars $k_g$ and $\gamma_c$ in equation (2.30). The function $\alpha(.)$ in equation (2.36) is taken to be zero for the sake of simplicity. These scalars are initially set to some small values and tuned on the nonlinear simulation model of the engine. Later on, they are tuned on the actual rig. $\gamma_c$ provides a very effective knob for improving the disturbance rejection properties. The investigations has shown that $k_g = 0.3, \gamma_c = 0.1$ improves the speed recovery rate without inducing any oscillations or input chattering.

In Section 2.4 it is assumed that the plant is square (i.e. the number of inputs and outputs are equal). However, in this ISC problem there are two inputs and one output. The square plant assumption is needed for observer design only. The first input channel, ABV, has most of the authority as compared to second input channel SPA, so the observer is designed for the first input only. This observation is supported by the fact that the steady state gain of the ABV channel to the engine speed is $4.66e03$ as compared to that of SPA channel of $-10.57$.

### 7.8 Sliding Mode Controller-Observer Implementation

To facilitate efficient controller tuning and testing, the sliding mode controller-observer pair is implemented in the form of a rapid prototyping framework, which enables the user to design and test the controller on the car engine in a matter of few minutes.

#### 7.8.1 Rapid Prototyping Tools

Sliding mode controller-observer pair has a very sophisticated and involved structure, so its not possible to implement this controller with simple electronic components as is done for the PID controllers. Such a controller must be implemented by means of a high speed computer having input-output arrangements such as IO cards with A/D and D/A converters. If an onboard controller is required, then a dedicated microprocessor system should be designed. The controller is designed in a matrix solver environment and should be realised in a high level programming language like C. This high level controller code
is cross compiled to make a controller executable file. This file is downloaded onto the microprocessor, acting as a host to the controller. The microprocessor receives the output measurements from the system, computes the right plant inputs through the controller file downloaded and sends this information to the plant inputs. Considerable man-hours would be consumed if all these steps were performed manually. Besides, there is a high probability of coding errors. These factors result in high cost and low efficiency for controller testing.

In the market, many ‘off the shelf’ products are available which can serve as automatic rapid controller prototyping tools, enabling efficient controller testing and tuning. Two such compatible products are marketed by dSPACE GmBH (Digital Signal Processing and Control Engineering) and ADI (Applied Dynamics International). ADI provides a combination of different tools to constitute such a framework. These are EASY5 from the Boeing Company which may be used for the controller design and analysis, combined with BEACON for C code generation. The host processor’s name is ADRTS which is essentially a 100 MHz Power PC.

dSPACE provides such an integrated framework in conjunction with Mathworks. It uses MATLAB for the controller design, Simulink\(^1\) for the controller realisation and Real Time Workshop\(^2\) for the C code generation. This C code is cross compiled and downloaded onto the host microprocessor. Initially, this processor used to be Texas Instrument TMS320C30 or TMS320C40. Recently, a Digital Systems Alpha processor (300 MHz) is also employed as the host processor.

The cost of an average dSPACE system is at least ten times less than an equivalent ADI system. The speed of dSPACE system is markedly higher than that of ADI. These factors make dSPACE more preferable.

### 7.8.2 dSPACE

dSPACE consists of specific software and dedicated hardware to run Simulink structures on a microprocessor in order to interface Simulink (and in turn Matlab) to the real world. The block diagram (Figure 7.13) shows the working of dSPACE. The implementation

---

\(^1\)Product of Mathworks

\(^2\)Product of Mathworks
Figure 7.13: Controller prototyping in dSPACE
process consists of the following steps:

- The controller is designed and drawn in Simulink. Blocks depending on the absolute notion of time cannot be used, as they are not implementable in dSPACE. It should be attempted to keep the number of blocks to a minimum.

- The controller components which cannot be represented by Simulink blocks should be written in the form of S functions. The S function is a specific file format in the Matlab environment used to write and simulate dynamic systems. This S function should be written in C in the MEX file format. A MEX file is a kind of C format used in Matlab for external interface.

- Once the Simulink controller representation is completed, then it is converted into C code.

- This C code is further cross compiled into an executable file to be run on a Texas Instrument microprocessor TMS 320C30. This microprocessor is connected with the plant to be controlled.

- Now the controller is downloaded onto the processor and it controls the plant.

The controller-observer pair is essentially a nonlinear dynamic system to be realised in Simulink for dSPACE implementation. There are various ways to model a dynamic system in Simulink.

### 7.8.3 Dynamic System Modelling in Simulink

The various formats used for dynamic system realisation in Simulink are:

1. Simulink Block Diagram
2. M Function
3. S Function (as a feedforward function)
4. S Function
5. S Function (in C Language)
Chapter 7. Idle Speed Control Of An Automotive Engine

The Simulink block diagram method is the simplest method to model a dynamic system in Simulink. Using the standard Simulink block library, a wide range of nonlinear systems can be realised. Nonlinear differential equations can be realised by means of nonlinear blocks and integrator blocks. If the differential equations are linear then the linear state space block can be used. However, the continuous time state space block cannot be used as it is not compatible with the dSPACE software. The user cannot vary the step size of the numerical integration during simulation according to his will. This is a major problem for this study as the sampling time of an engine varies according to the engine speed. The speed of the observer will be limited by the Simulink step size which has to be the same as the engine step size. In addition, the method is very time consuming and it is difficult to make changes at the later stages.

The second way is to use an M function block. The behaviour of this block can be defined by the user in the form of a Matlab script file or 'm file'. The integration routine has to be written inside the script file for solving differential equations. The advantage lies in the control of the integration parameters. The user can easily vary the step size according to the engine speed. Similarly, s/he can choose faster observer dynamics by decreasing the integration step size for the observer. However, to write such a script file one needs considerable experience in Matlab. This M file is not directly implementable on dSPACE. It has to be translated into a C program with a very specific format called MEX S function files, which is used to interface Matlab with C routines.

S functions (System Functions) are a versatile way of simulating dynamic systems. They are Matlab script files written in a very specific format. This is the most efficient tool for modelling nonlinear differential equations in Simulink. It suffers the same problem as the Simulink block diagrams i.e no control over the step size. However, if the S function is written as a simple input output block (called feedforward mode), this problem can be solved. In order to implement the S function block into dSPACE, it has to be converted into the MEX S function files, which is very time consuming. There is no good tool for debugging a MEX file. Also, Matlab is very sensitive to dynamic memory allocation errors. If such errors are present in the MEX file, then it can crash the whole Matlab session.

In essence, there are two extreme ways of implementing the controller in Simulink/dSPACE.
Chapter 7. Idle Speed Control Of An Automotive Engine

The first involves making a very detailed block diagram of the whole controller-observer structure while the second involves writing the whole controller code in the C language in the format of a MEX S function file. The first method will give a very specific solution while the second method results in a very complex controller implementation code. Here a method in between these two extremes is chosen. Most of the linear controller-observer components are absorbed into one big state space formulation and this is implemented through a MEX S function written in C, which is not very complex. The nonlinear part is realised through the nonlinear blocks from the Simulink Nonlinear Block Library. This realisation is not very difficult to understand and, at the same time, it is quite generic.

Another important issue is that of the parameter transfer from the controller design environment (Matlab) to the controller implementation environment (dSPACE). The usual way of passing parameters in such situations is through data files. In the case of the Multiburner furnace controller design (Chapter 6), the controller is coded manually in Fortran. At the start of the furnace test simulation, the controller routine reads the controller parameters from the data files, generated by the controller design routine in Matlab. However, when the controller is run from a microprocessor, no data reading facility is available. In this case the controller realisation is done in such a way that Simulink (and dSPACE) directly reads the parameters from the Matlab workspace. The design routine thus provides all the parameters to the workspace instead of storing them in a data file.

7.8.4 Controller Realisation

The observer is translated into a MEX file written in C. The controller structure in Simulink is shown in Figure 7.14. From the discussion in the previous section it is clear that the main controller blocks are:

1. Observer

2. Nonlinear control component

3. Linear control component
A Simulink realisation of the overall controller is shown in Figure 7.14. The observer can be sub-divided into linear and nonlinear components. The linear component of the observer state space system is encoded through a MEX file S function written in C. All other dynamic computations such as the demand dynamics equation have been absorbed into the observer linear part. The detailed outline of the observer structure is shown in Figure 7.15. The CMEX S function simulating the linear dynamic part of the observer is called Isys1. Figure 7.16 shows the block for computation of $\rho_\sigma$ as described in the previous sections. The linear component is implemented using a standard Simulink gain matrix block. The nonlinear control component (Figure 7.17) computes $\rho_c$ and in turn $u_n$.

7.9 Linear State Space Formulation

As mentioned in the previous section, the linear state space dynamics are absorbed in the block Isys1. The overall state space system will be explained here. All the terminology used here is consistent with previous sections. Consider a state space construction with
Chapter 7. Idle Speed Control Of An Automotive Engine

Figure 7.15: Observer realisation

Figure 7.16: $p_o$ computation

Figure 7.17: Nonlinear Control Component
the input $U \in \mathbb{R}^{p+n+m}$:

$$U \triangleq \begin{bmatrix} W \\ y \\ u \end{bmatrix}$$

(7.19)

where $W \in \mathbb{R}^p$ is the unfiltered demand (appearing in equation (2.10)), $y \in \mathbb{R}^p$ is the plant output and $u \in \mathbb{R}^n$ is the plant input. The output of this state space construction is defined as the state vector $X \in \mathbb{R}^{p+n+p}$ of the state space system with state

$$X \triangleq \begin{bmatrix} z_e \\ z \\ w \end{bmatrix}$$

(7.20)

$z_e \in \mathbb{R}^p$ (equation (2.12)) is the integral of the output error, $z \in \mathbb{R}^n$ is the state vector of the linear system in regular form (2.9) and $w \in \mathbb{R}^p$ is the filtered version of the demand governed by equation (2.10). The state space matrices are formulated as:

$$A_{big} = \begin{bmatrix} 0 & 0 & KI_p \\ 0 & A_{obs} & 0 \\ 0 & 0 & \Psi \end{bmatrix}$$

$$B_{big} = \begin{bmatrix} 0 & -KI_p & 0 & 0 \\ 0 & O & B_x & B_{obs} \\ -\Psi & 0 & 0 & 0 \end{bmatrix}$$

(7.21)

The output distribution matrix $C_{big} \in \mathbb{R}^{p+n+p}$ is an identity matrix. In a similar way matrix gains are also lumped together to give a simplified Simulink realisation.

### 7.10 Forming an Inner Loop

It is observed during the nonlinear simulation that the Spark Channel (SPA) has very low control authority, thus tending to saturate very quickly. Once SPA has saturated it degrades the controller performance. This constrains the speed of the controller. An alternative to avoid this situation is to use an inner loop with proportional gain on SPA.
Chapter 7. Idle Speed Control Of An Automotive Engine

Figure 7.18: Closing the spark channel by the proportional gain

and use the new construction for sliding mode controller design. The modification in the original system by closing the spark loop is discussed next.

The nominal linear model of the engine is characterised by the triplet \((A, B, C)\) such that:

\[
B = \begin{bmatrix} b_a & b_s \end{bmatrix}
\]

(7.22)

where \(b_a\) and \(b_s\) are the two columns of the input distribution matrix \(B\) with \(b_s\) pertaining to the spark channel. If \(p\) is the proportional gain used to close the negative feedback loop on the spark input, then the new system matrix \(A_{\text{new}}\) is:

\[
A_{\text{new}} = A + pb_s(-C)
\]

Hence the new system triplet will be \((A_{\text{new}}, b_a, C)\), with the air bypass valve as the only input. This triple is used for sliding mode observer/controller design. The new controller arrangement is shown in the figure 7.18. The results of the controller with and without inner loop on the nonlinear model provided are discussed in the next section.

7.11 Nonlinear Simulation Tests

After describing the controller design and formulation, the nonlinear simulation results are presented. First the results of minimum entropy design are shown. Later on, the
Chapter 7. Idle Speed Control Of An Automotive Engine

Step changes in load at 880 rev/min

Figure 7.19:
simulation results of the controller with the inner loop on SPA are illustrated. The results of this and subsequent Sections are given in the appendices. The controller designed in Section 7.7 is tested on the nonlinear simulation model. A whole range of simulation tests suggested by the Ford Motor Company have been performed on the given nonlinear simulation model. A representative simulation test is shown in Figure 7.19. As noticed from this figure, the dips are quite low. The speed of the MIMO controller cannot be increased at will as the SPA input channel has very low authority. This implies that the full potential of the ABV channel cannot be utilised. In order to increase the speed of response, a proportional loop on the SPA channel is constructed with a gain of 0.1 (details are in Section 7.10) and the sliding mode controller is designed only for the ABV channel. Now the response of the controller can be increased without worrying about the saturation of the SPA channel. The design parameters remain as before with the exception of the hyperplane design. Now the minimum entropy design is accomplished using the following variables.

\[
Q = \begin{bmatrix}
1e0 & 0 & 0 & 0 & 0 \\
0 & 1e-4 & 0 & 0 & 0 \\
0 & 0 & 1e-4 & 0 & 0 \\
0 & 0 & 0 & 1e-3 & 0 \\
0 & 0 & 0 & 0 & 1e-4
\end{bmatrix}
\]

\[
R = \begin{bmatrix}
1e0 \\
\end{bmatrix}
\]

\[
W = \begin{bmatrix}
1e-2 & 0 & 0 & 0 & 0 \\
0 & 1e-2 & 0 & 0 & 0 \\
0 & 0 & 1e-2 & 0 & 0 \\
0 & 0 & 0 & 1e-2 & 0 \\
0 & 0 & 0 & 0 & 1e-2
\end{bmatrix}
\]

\[
V = \begin{bmatrix}
1e-1 \\
\end{bmatrix}
\]

\[
K = 3.0
\]

The weightings appear to have slower dynamics than that of the first controller (7.19) but \( K \) is increased to 3 thus making the controller faster. The controller yields the condition number of 55.88, which is lower than its MIMO counterpart (94.16). This
controller is tested on the nonlinear simulation model. A representative of the nonlinear simulation results is shown in the Figure 7.20. From the inspection of the corresponding figures it is obvious that the dips have been improved considerably but with a significant deterioration in performance during the robustness tests. After the controller testing on the nonlinear simulation model, the controller is tested on the rig, as explained in the next section.

7.12 Rig Results

This chapter explains the rig results for both MIMO and SISO controllers. As previously described, the controllers should exhibit good tracking, robustness and disturbance rejection properties. On the test rig these properties of the controller are tested. These tests are described in the next sections.

7.12.1 MIMO Controller Results

In this section the results pertaining to controller without the inner loop are shown. The tests can be categorised as:

1. Performance tests
2. Robustness tests

Performance Tests

The setpoint speed is chosen to be 880rpm. The tracking and disturbance rejection properties are verified by applying different loads on the engine and observing the controller performance. From Figure A.4 it is observed that the controller is keeping the speed within the band of 20rpm despite the disturbance generated due to stochastic combustion. For the dips, the controller is returning the engine to setpoint speed at a rate similar to that of descent. The overshoots are less than 4%. For the case of flares, the magnitude of speed is not exceeding the 10% of the setpoint speed. The maximum dip is around 17%. This dip can be reduced, but at the cost of increased oscillations, high overshoots and lower robustness.
Figure 7.20:

Step changes in load at 880 rev/min
Robustness Tests

The robustness of the controller can be highlighted by considering the fact that the controller design is performed at the speed of 720 rpm. However, these tests are performed at a speed of 880 rpm. A further disturbance rejection test is performed with different gains and time delays included. The controller easily stabilised these load tests with a delay of five sample intervals and the gain increased by a factor of 1.5. Figure 7.21 shows a performance test for various loads and accessories, applied and withdrawn at the marked points A to H. The same test is performed with a delay of 5 sample periods on the measured output. The results are shown in Figure 7.22. By comparing Figures 7.21 and 7.22 it is seen that despite the heavy disturbance due to the delay the controller is still maintaining reasonable performance.

7.12.2 SISO Controller Results

In this section the results pertaining to the controller with the inner loop are presented. Figures B.1 to B.8 show these results. In Section 7.11 it has been observed that the formation of an inner loop improves the controller performance to a considerable extent, especially as far as the dips are concerned. It is hoped that the SISO controller would exhibit similar properties on the test rig. However, by comparing the corresponding results for the MIMO and SISO controllers, it is revealed that the same degree of improvement could not be achieved.

It can be concluded that the SISO controller is showing good steady state response with quite low controller effort and reasonable robustness. The only area where this controller is not performing particularly well is that of reduction of dips. The dips can be reduced on the rig by increasing the speed of the controller, which will further result in the input channel saturations, decreased robustness and hunting of the engine, thus aggravating the overall engine performance. In the industry, feedforward controllers are used to overcome the excessive dip problems. This sliding mode controller has a provision for incorporating such a feedforward scheme in the overall controller framework. The initial investigations and the nonlinear simulation results show promising results. This area needs further research.
Figure 7.21: Rig Results for Test Three
Figure 7.22: Rig Results for five sample delays


7.13 Direct sliding mode controller design

After the linear model based sliding mode controller had been tested on the engine rig, a direct sliding mode controller based on the parameter estimation technique proposed in Chapter 5 is designed. This design is for single input only i.e. air bypass valve input. As in Section 6.12 the design is done in two phases.

In the parameter estimation phase model parameter estimation is performed. A two state model is used with states consisting of the output temperature and its first derivatives. The parameter estimation is performed at maximum load. Figure 7.23 shows the parameter convergence. The state error sliding surface is characterised by the Hurwitz polynomial $[1 10]^T$.

The second phase consisted of the controller design. The controller is designed on the basis of the model (5.1) with estimated parameters with the following gains:

$$k_c = 10.0$$
$$k_d = 10.0$$

These gains are selected on the basis of observation, getting a compromise between good performance and chattering.

The simulation test (Figures 7.24 and 7.25) shows the robustness of the controller which is tracking setpoint speed of 720rpm. The loading is changed at 1.0 s, causing the dip in the speed which is recovered soon. The Hurwitz polynomial specifying the sliding surface is $[1 10]$. One point should be noted which is significant from the implementation point of view. Due to nonlinearities present, a suitable sample time is 3.0 ms as compared to 30.0 ms which is used for linear model based controller implementation. This can cause a serious limitation if there is no control over sampling rate during controller implementation, as is the case for the controller tested on the rig.

Now a new set of simulation tests is performed to compare linear model based controller (Figures 7.26 and 7.27) with DSM controller (Figures 7.28 and 7.29). The linear model based controller is slower because of the demand filter used in its construction which is not the case with DSM controller. The linear model based controller also shows overshoot in the transient which is not present in the DSM controller simulation.
Chapter 7. Idle Speed Control Of An Automotive Engine

Figure 7.23: Engine model parameter estimation
Chapter 7. Idle Speed Control Of An Automotive Engine

Figure 7.24: DSM controller performance

Figure 7.25: DSM controller output and sliding surface
Chapter 7. Idle Speed Control Of An Automotive Engine

Figure 7.26: Linear model based controller performance

Figure 7.27: Linear model based controller output and sliding surface
Chapter 7.  Idle Speed Control Of An Automotive Engine

Figure 7.28: DSM controller performance

Figure 7.29: DSM controller output and sliding surface
7.14 Summary

This chapter has presented a practical linear model based nonlinear controller design method for automotive engine speed control. The Chapter has started with a description of the control problem and the approaches used in the past to solve the ISC problem. Then the available discrete time models of the engine have been converted into continuous time models. The open loop characteristics of the engine have been inferred from these models. A detailed analysis have been performed to select the model for the controller design. The scheme explained in Chapter 2 has been decomposed into controller design steps and the implementation of each step has been discussed. Two methods for the hyperplane design have come under discussion. Their robustness is compared by highlighting the relationship between their cost functions. After discussing these general issues, the sliding mode controller-observer has been designed. Results of the different controller designs have been compared. For the controller implementation, various rapid prototyping tools have been reviewed. Later on, the working of dSPACE has been described in detail. The controller realisation has also been discussed. Various issues like, integration step size, controller complexity and generic nature of the realisation have been kept in mind before deciding upon the right realisation format. The linear state space formulation has been proposed for the controller implementation, followed by results of nonlinear simulation testing of the controller. This leads on to the successful testing of the controller on the test rig. Since the controller is designed in a multivariable framework, and assumes only a subset of state information is measurable, it is applicable to other automotive control problems such as air to fuel ratio control. The controller shows good performance and robustness, hence rendering it suitable for practical application. In addition, the validity of the design technique proposed in Chapter 5 is also demonstrated by using this method for controller design for the ISC problem. The next Chapter concludes the thesis.
Chapter 8

Conclusions And Future Work

8.1 Concluding remarks

The various phases of industrial application of sliding mode controllers are explored and demonstrated through controller design for different industrial problems. First the theory regarding linear model based sliding mode controller design based on the work of Edwards and Spurgeon [20] is presented. Using this method, a controller design procedure for an industrial controller, starting from modelling, is developed. The effective use of various degrees of freedom in the controller design is explored. Direct hyperplane design using various approaches is proposed and integrated into the design framework. The sliding mode controller design when the model is nonlinear is also explored. Two generic design techniques, DSM and ISM, proposed in [49, 47, 50] are discussed. Their design procedures are described in detail. In order to apply these techniques, the need for the GCCF representation of a given industrial process is discussed. Two ways are suggested to solve this nonlinear identification problem.

Feedforward networks are used to identify the nonlinearity in the system, hence determining a GCCF model for the system to be controlled. The DSM controller needs uncertainty bounds for its gain computation. A backpropagation technique with robust cost function is proposed. Various ways to establish the uncertainty bounds are given. Through design examples, these bounds are verified.

The second way to determine the GCCF representation of a nonlinear system is to assume a nonlinear model with known nonlinear structure but unknown parameters.
Both off-line (open loop) and on-line (closed loop) parameter estimation is investigated. The stability is proved using quadratic stability concepts.

The above mentioned design procedures are applied on two challenging industrial processes. The application is not merely a demonstration of the validity of the controller design methods discussed but the solution of these industrial control problems is a novel contribution in itself. It has been shown that the linear model based method (Chapter 2) and sliding mode based parameter estimation method (Chapter 5) worked effectively for industrial problems. The neural network model based method (Chapter 4) does not show very promising results. The controller shows excessive chattering because of the uncertainties in the neural model identified.

Multiburner furnace temperature control is a challenging control problem with very strong coupling and nonlinearities. The existing control literature suggests that this area is still to be explored. Two sliding mode controllers are designed and tested successfully for such a furnace.

The difficult problem of idle speed control of an automobile engine is solved through sliding mode control. Successful rig tests are performed to substantiate the claim. A generic dSPACE based sliding mode controller is constructed and reported.

The next section considers some possible research directions for future work.

### 8.2 Future Work

The future research directions originating from the research reported in this thesis are:

- Recalling the GCCF model representation from Chapter 3:

\[
\begin{align*}
\dot{z}_1 &= z_2 \\
\dot{z}_2 &= z_3 \\
& \vdots \\
\dot{z}_{n-1} &= z_n \\
\dot{z}_n &= \eta(z, \dot{u}, t)
\end{align*}
\]

where

\[
z = [z_1, z_2, \ldots, z_n]^T = [y, \dot{y}, \ldots, y^{(n-1)}]^T
\]
. The first state $z_1$ is the system output and all the other states are its successive derivatives. For the modelling schemes discussed in Chapters 4 and 5 need all these states for their modelling procedure. In other words, the system output and its $n$th derivatives should be known or measurable. In most cases the output would be known but not its higher derivatives. This can cause a serious limitation for the modelling and the controller design process. Normally, this problem is circumvented by generating the output derivatives numerically. Such derivative computations results in noisy derivative signal estimation. The noisy signals will affect neural network modelling as well as the parameter estimation scheme. In addition, if the controller is implemented with numerically computed derivatives, this may result in a high frequency component in the control signal, thus, resulting in input chattering. Of course, this high frequency component can be eliminated with low pass filtering but this solution would be very problem specific.

A proper generic solution would be to use a nonlinear observer but a generic nonlinear observer does not exist. In the literature people have used high gain observers for the output derivatives estimation [38]. So a high gain observer can be used to estimate the derivatives of the output, as it does not need the system structure for derivative estimation. A useful area of research could be to investigate the stability of a high gain observer working with a DSM controller, or even to have a parameter based estimator with a high gain observer. Similar work for a state feedback controller with a high gain observer has been reported by [38]. Of course, in the case of DSM the analysis will be more involved.

- Another interesting area of work would be to extend the parameter estimation scheme introduced in Chapter 5 into an adaptive observer. So that the scheme can estimate both states and the perimeters at the same time. The study on this problem has shown that, using sliding mode concepts, such an observer can be formulated. But this formulation needs a very strong assumption that the output should be a subjective function of the system states i.e for every combination of the states there should be a unique output. This is a very strong assumption for multidimensional system. This problem can be investigated further by looking into finding such functions or using any other alternate approach to tackle the
Chapter 8. Conclusions And Future Work

problem.

• The research in the parameter estimation of nonlinear models has concentrated on the case when the parameters are appearing linearly [38, 89]. The reason lies in the fact that the linear appearing parameters can be directly dealt with in Lyapunov stability analysis. An investigation of the problem has revealed that using sliding mode concepts, this problem can be tackled. A modified version of the model 5.1 with the \( \Gamma_n(.) \) function changed to a known nonlinear function of the states \( x \) and the unknown system parameters \( W \):

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 \\
&\vdots \\
\dot{x}_{n-1} &= x_n \\
\dot{x}_n &= \Gamma_n(x, W) + pu + \Delta(x, t) 
\end{align*}
\tag{8.2}
\]

where \( x \in \mathbb{R}^n \), \( W \in \mathbb{R}^n \), \( u \in \mathbb{R} \) is the control input and \( p \in \mathbb{R} \) is the input gain. \( \Delta(x, t) \) is the model uncertainty with a known bound. In order to devise a sliding mode parameter estimator for the model 8.2 certain assumptions on the function \( \Gamma_n(.) \) have to be made. The function should be Lipschitz with respect to both \( x \) and \( W \). The first derivative of \( \Gamma_n(.) \) should exist and the unknown parameters should appear linearly in the first derivative. Furthermore, the uncertainty and its first derivative should be bounded with known bounds. Using these assumptions, the convergence of the estimated parameters can be shown. However, there are a few problems. The assumptions on \( \Gamma_n(.) \) are too many. Besides, substantial amount of work is needed to show that the scheme works for real systems.

• The modelling scheme in Chapter 4 has only exploited feedforward neural networks. Initial investigation has shown that another popular kind of neural networks called Radial Basis Function Networks (RBFN) can also be used in the modelling scheme instead of FFNN. Though, the structure of RBFN is a bit more complex [29].
Modelling techniques like FFNN or RBFN are essentially static in nature. They do not have hysteresis property. However, Recurrent Neural Networks (RNN) or sometimes called Dynamic Neural Networks (DNN) are essentially dynamic in nature. Since, in these structures, the network output is also feedback to the network inputs. Rovithakis and Christodoulou \([67, 65, 66]\) proposed a dynamic neural network structure and a training algorithm for such a system. The DNN model is \([66]\);

\[
\dot{x} = -Ax + WS(x) + W_{n+1}\tilde{S}(x)u
\]  

(8.3)

where \(W\) are the unknown parameters or weights and \(S(x), \tilde{S}(x)\) are the sigmoidal functions. Further details of the model can be found in \([66]\). But the point to emphasise here is that the model (8.3) is not in GCCF form. Work can be done to transform (8.3) into GCCF representation and then use it for DSM design. Alternately, the possibility of a sliding mode controller design method for a system representable in the form of (8.3) can be probed.

The most common approach in neural network modelling is to obtain a discrete time model which predicts the next output based on the knowledge of past and present inputs and outputs. The research in this thesis has dealt with the control of the systems representable in neural networks based continuous models. A logical extension to this work would be to look into the ways of formulating a sliding mode controller for a neural network based discrete time model. Utkin and Yu have looked into the problem of discrete time sliding mode control \([109]\). There approaches can be used as starting point for this investigation.

In Chapter 5 the structure of the nonlinear model is assumed to be known. A further step ahead would be to investigate various ways of finding out the nonlinear functions \(\Gamma(.)\) for a given class of physical systems. Ideas about a possible structure can be extracted from nonlinear function approximation methods like Radial Basis Functions or Polynomial Approximations.

Successful controller design and testing for the temperature control of multiburner furnaces is reported in Chapter 6. A logical extension to the work is to implement such a controller on an actual multiburner furnace. The simulation code and
controller design framework considered is fairly generic and is thus applicable to a wide variety of gas furnaces.

- The furnace nonlinear simulation code is written in Fortran which is an extremely complicated collection of many subroutines thus making further modification and debugging difficult. Being in Fortran, every new control scheme designed need to be coded into Fortran and interfaced with the furnace simulation code for controller testing. This step gets further complicated when the controller is extremely nonlinear and dynamic. In this case numerical integration routines in Fortran are required to compute the controller outputs. A possible suggestion could be to use the NAG libraries. But the variable definition incompatibilities between the simulation code and NAG libraries ODE solvers makes the use of such libraries less desirable. Since the controller design would be done in a matrix solver environment like Matlab, it would be efficient to have the furnace simulation model in Matlab or Simulink. It can be attempted to convert the simulation code into the MATLAB/SIMULINK environment, hence making it easily accessible for controller design. This conversion can be easily done due to the addition of GTPOWER block library to Simulink. GTPOWER is a set of tools to solve partial differential equation, which would be extremely useful in simulating the furnace model. Another point in favour of this work proposal is the complexity of a nonlinear controller resulting from a symbolic computational software used for the controller design. It may be very difficult to code such a controller structure and its parameters in a time efficient way. If the symbolic computations are performed in Matlab or Maple then they can be easily transferred to a Simulink model. It can be argued that a Fortran model will run faster than its Simulink counterpart, but a variety of tools do exist which can automatically generate independently executable C code for a given Simulink model. The execution speed of the generated C code will be compatible to the written Fortran code.

- Linearly identified models were used for the sliding mode controller-observer design in Chapter 7. The linear identification methods suffer from the fact that the states of the linear identified model does not have any physical interpretation, which is a drawback when weighting states in controller design methods like LQ or ME
design. A possible extension of the current work could be to establish automotive engine models on the first principle basis, linearise them and utilise them for the sliding mode controller-observer design. A good approach on automotive engine modelling can be found in Kao and Moskwa's work [36]. A practical modelling approach for automotive powertrain modelling can be found in the work by Cho and Hedrick [95].

Chapter 7 has only addressed the problem of the engine speed control. A whole range of problems exist in the automotive field which can be solved by using the methods described in this thesis. Strict regulations on pollution control motivates the need for tighter emission control in automobiles. The emissions in an automotive engine depends heavily on how far is the actual air to fuel ratio from the stoichiometric air to fuel ratio. Hence, the emissions control problem can be controlled by keeping air to fuel ratio control close to the stoichiometric ratio. In spark ignition engines this can be achieved by manipulating air intake into the engine cylinders while in compression ignition engines the fuel quantity injected into the cylinder can be a possible control input. Won and associates [100] address this issue with sliding mode control, which motivates the need for air to fuel ratio controller design using the techniques introduced in this thesis. As the techniques are multivariable in nature, they can be used to consider ISC and air to fuel ratio control in one integrated framework. Similarly the speed control of a vehicle by manipulating air or fuel inputs to its engine is also an interesting control problem which can be tackled using the design method described in Chapter 7.

The testing of the ISC controller was performed for one speed only and also for a very limited speed range. A further course of research could be to design a controller for the entire engine speed envelope. This controller can be used as a lower level controller in an a bigger control loop of cruise control in which the cruise controller is expected to maintain the speed of the car at a desired value. Since, the idle speed controller designed is good for tracking and disturbance rejection, it will be useful for cruise control strategies.
Chapter 8. Conclusions And Future Work

- The sliding mode observer can be used for the robust state estimation of the states of a physical model of the engine system. Using such an observer, valuable information like the engine cylinder pressures, engine torque, load torque and air to fuel ratio can be estimated. These estimations will provide the information which will be very useful for engine control, monitoring and fault diagnostics. Kao and Moskwa [35] estimated air to fuel ratio and the load torque through sliding mode observers for a particular automotive engine, but their observer construction was very specific to the engine considered.

- The load accessories on an automotive engine are considered as unknown disturbances. Thus the disturbance rejection strategy through the sliding mode controller is a pure feedback strategy. In the automotive industry, it is possible to know the kind of load accessory being applied at a given time. So a feedforward control is applied to regulate speed with respect to each disturbance. This feedforward control is essentially a combination of feedforward gains on the ABV (air supply) scheduled according to the nature of the accessory applied. It is proposed to determine transfer functions of each disturbance to the engine speed and then utilise it in the sliding mode controller to get further reduced dips.

- It can be attempted to model the uncertainty considered in Chapter 2 (equation 2.8) using neural networks. The residuals of the linearly identified models can be used as the input output data for the neural network training. It is possible to determine the bounds on the uncertainty by analysing these networks.

- The sampling rate in the automobile rig was fixed to be two times every revolution, thus limiting the controller speed. This limitation can be avoided by looking into arrangements for increasing the sampling rate of the engine. By increasing the sampling rate, their will be lesser time delays and more controller updates.
Appendix A

Rig Trial Results of MIMO controller
Appendix A. Rig Trial Results of MIMO controller

Figure A.1: MIMO controller results
Figure A.2: MIMO controller results
Appendix A. Rig Trial Results of MIMO controller

Figure A.3: MIMO controller results
Figure A.4: MIMO controller results
Appendix A. Rig Trial Results of MIMO controller

Figure A.5: MIMO controller results
Figure A.6: MIMO controller results
Figure A.7: MIMO controller results
Appendix A. Rig Trial Results of MIMO controller

Figure A.8: MIMO controller results
Appendix B

Rig Trial Results of SISO controller
Appendix B. Rig Trial Results of SISO controller

Figure B.1: SISO controller results
Appendix B. Rig Trial Results of SISO controller

Figure B.2: SISO controller results
Appendix B. Rig Trial Results of SISO controller

Figure B.3: SISO controller results
Figure B.4: SISO controller results
Figure B.5: SISO controller results
Figure B.6: SISO controller results
Appendix B. Rig Trial Results of SISO controller

Figure B.7: SISO controller results
Appendix B. Rig Trial Results of SISO controller

Figure B.8: SISO controller results
References


References


