REAL TIME FAULT DETECTION AND DIAGNOSIS IN DYNAMIC ENGINEERING SYSTEMS USING CONSTRAINT ANALYSIS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING

OF GLASGOW UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

N. A. Marrison

October 1992

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Abstract

This thesis describes some new ideas and a practically orientated implementation for fault detection and diagnosis in dynamic engineering systems. The method is designed for use on-line, it is model based, and is capable of coping with modelling inaccuracies, noisy measurements from the system and unmeasurable system states. The fault detection system is robust to false alarms, and the fault diagnosis system allows for the possibility that multiple faults may occur simultaneously.

A number of system analysis algorithms are presented to extract various system equations from the model of the system. This means that the user need only enter one model of the whole system, and all of the analysis and equation solving is then handled by computer. The results of this analysis are then automatically encapsulated into a fault detection and diagnosis tool. This results in the automatic generation of a specific fault analysis tool for the system entered by the user.

A "hypothesis prover" is developed here for the domain of dynamic systems, which is used to test hypotheses. Some of the ideas about multiple faults as developed by de Kleer & Williams and Reiter have been used, but these have been adapted to make them applicable for real-time, recursive, imprecise, diagnosis. (Diagnoses are imprecise because, due to modelling errors and noisy measurement, it is never possible to be 100% certain about anything.)

When multiple faults are considered, the number of possible combinations becomes very large, $2^N - 1$, where N is the number of components. The computation required to prove a particular hypothesis, although not enormous, is not trivial either, making it impractical to prove a large number of hypotheses. To overcome this a method is proposed which involves just proving a subset of the possible hypotheses, and using the information obtained from these to reason about the other hypotheses. This requires much less computational power as the reasoning process is much less intensive than the proving process. This make the diagnosis of multiple faults possible in real-time.

The methods developed here are tested on a real, noisy system where approximations are made when producing the systems' model. These tests show the potential of this approach to fault diagnosis.

Preface

Fault detection and diagnosis in dynamic engineering system has received much attention recently. New and original approaches to this area have been made possible by the advances in computing technology which allow fault analysis tools to exploit analytical and knowledge-based redundancies that were not possible before.

In particular the "*Theory of Diagnosis from First Principles*" by Reiter [68] has laid the way for new developments in the diagnosis of multiple faults and competing fault hypotheses.

This thesis describes my attempt to use some of these powerful ideas, together with some new extensions and apply them to the domain of dynamic engineering systems. A description of how they could be applied in a practical sense is given.

Acknowledgements

I would like to thank my supervisor, Professor Peter J Gawthrop, for the support, advice, trust and encouragement throughout this work.

I would also like to thank all members of the Control Group in the Mechanical Engineering Department of Glasgow University for the friendly and supportive atmosphere, and in particular Dr Donald Ballance for his help and patience while I was discovering the secrets of Unix.

This work was made possible by the Engineering Design Research Centre which I would like to thank for the excellent facilities and support which was given to me.

I thank my mother and father for the help and support they have given me during this work.

Finally I would like to thank Xenia Richter for her help in organising my work, and for her continual encouragement, support and understanding.

Contents

List of Tables

List of Figures

Chapter 1

Introduction.

This thesis describes some new, practical, model-based fault detection and diagnosis methods. These methods are capable of detecting and diagnosing single and multiple faults in dynamic engineering systems, even when noise is present in the system's measurements, and the model of the system is imprecise. In addition to this a method for automatically generating a portable and complete implementation of the fault analysis algorithms for a specific system is described.

1.1 Motivation

The need for improved performance in industrial situations has caused a need for improved fault detection and diagnosis methods. This, together with the advances in computing technology, has resulted in a whole new range of detection and diagnosis tools which are both possible and practicable.

Detecting a fault in a system does not just mean discovering that the system has failed, but also discovering a component of the system which no longer performs to it's specification. This could be a bearing that has worn, a pipe which is becoming clogged or some thermal insulation which has deteriorated. This degradation in a component's

performance needs to be detected and pin-pointed, as if it is left unattended, a gradual reduction in the system's performance is inevitable.

The earlier a change in the system's performance can be detected and diagnosed, the earlier preventive maintenance can be carried out, thus helping to ensure that the required performance of the system is maintained.

In the past, fault detection has been achieved with the use of many plant sensors measuring many different quantities and ensuring that all of them are between specific limits. Attaching many sensors to a plant is obviously expensive, and the more sensors there are, the larger the possibility of a sensor failure and therefore a false alarm. There are also certain quantities which may be difficult to measure. One result of this is that a fault may not normally be detected until the system's performance substantially changes. The requirements for a fault detection system must therefore be to keep the number of sensors on the system to a minimum, and to be as quick and sensitive as possible in detecting faults whilst minimising the number of false alarms.

Fault diagnosis is often an imprecise science, depending upon an operator's understanding of the plant to determine where the fault is. A fault diagnosis system should be capable of accurately determining which components of the system are misbehaving, and also by how much. Since it is never possible to be 100% sure which is the correct hypothesis (due to system noise and modelling inaccuracies), it is necessary to determine which are the most likely hypotheses, rather than producing one hypothesis as being correct.

In addition to this it may be possible that if some slight degradation in the plant occurs, it may not be necessary to shut the plant down and fix it, but it may be possible to compensate for the degradation and maintain the system's performance. The operator should of course be made aware of the situation and give the go ahead for compensation to take place, making a note of the degradation for maintenance at

some point in the future. To do this requires the quantification of the new behaviour of any component of the system which has altered.

Although many fault detection and diagnosis methods have emerged, many of these have been designed for the off-line analysis of data to determine when a fault occurred and where it was. The methods described here are designed to work on-line. If a diagnosis algorithm is to be used on-line, it must also be practicable, i.e. it must be capable of being executed in real-time on a computer. Finally, many diagnosis methods produced require a significant amount of work to implement the methods for each new system that they are applied to; this can limit the usefulness of these methods to only substantial systems which are worth the large amount of effort required. Any reduction in the effort required to implement fault analysis methods on new systems is obviously beneficial. For this reason, the task of analysing the system model and producing the necessary computer code to perform fault detection and diagnosis on a particular system has been automated. To facilitate this, the user is required to represent the dynamic system using bond graphs.

The detection and diagnosis methods given here are primarily concerned with handling the degradation of a plant, but they will also work in the event of a sudden failure of a component. The methods are designed for use on-line, in real-time, and incorporate a substantial amount of automatic system analysis and code generation.

Many detection and diagnosis systems require a large amount of tailoring to individual systems. This is one area where the methods described herein are trying to get away from. Here algorithms are presented which will automatically analyse many dynamic systems and extract various aspects of the systems behaviour which is then incorporated into a universal fault detection and diagnosis scheme. The user must enter the model of the system and the constitutive relationships for the individual components. After that a fault analysis tool for the system, which consists of a C++

program specific to that system, is produced automatically. This program contains all of the detection and diagnosis algorithms and the system equations needed. Much of the analysis of the system is done during this process, resulting in a final code which needs to perform only a minimal amount system analysis. It is therefore much more practicable with regards to execution speed, when used for on-line, real time detection and diagnosis.

The areas of original contribution contained in this thesis are:-

1. The adaptation of multiple fault diagnosis methods to a recursive in time, on-line situation for dynamic systems. That is, the conclusions of the diagnosis algorithms improve with time, the methods are practicable in real-time and it is not necessary that every state of the system is measurable.

2. Reasoning about sets and subsets of multiple fault hypotheses in situations where definite information about the correctness of a hypothesis is not available. (e.g. it is possible to say a hypothesis is likely to be correct or is unlikely to be correct, but one cannot say it is definitely correct or definitely incorrect.) This makes it possible to reduce the number of computationally expensive calls to the "hypothesis prover", and therefore increases the real-time practicability of the approach.

3. Using constraint propagation algorithms to analyse a dynamic system. These are unusual in that they may propagate signals through different periods of time (different sample times). The algorithms are also aware of algebraic loops and they will therefore always ensure (where possible) that there are enough equations to find the required unknown(s). That is the algorithms will propagate through different time periods to ensure that there is enough information to find the required unknown, but the propagation route will always be the shortest one possible.

4. Automatically generating a specific implementation of the fault analysis algorithms for a specific system. The implementation produced is self contained and is easily ported to a different platform from the one which produced it.

1.2 Overview

This thesis is in three main parts.

1. A discussion of survey material. (Chapters 2 and 3).
2. New developments and theories. (Chapters 4, 5 & 6).
3. Results of tests with simulated data and data from a real system. (Chapters 5, 6 & 7).

Chapter 2 contains a survey of work done in the fault detection and diagnosis area. This includes both fundamental work on the theory of faults, and specific implementations of some methods which have been developed. Methods for system modelling are also examined as is the background of the constraint propagation techniques.

Chapter 3 examines, in detail, the theory behind fault detection and how this has been done using constraint satisfaction. Fault diagnosis using constraint suspension is then discussed together with the circumstances when this is practicable. Examples of detecting and diagnosing faults are looked at for various types of system, and some of the problems associated with considering the possibility of multiple, simultaneous faults are described.

Chapter 4 looks in detail at the nature of dynamic systems and highlights aspects and features which one must be aware of to successfully develop a working system. A method is developed for examining dynamic systems when not all of the states

are necessarily measurable. Problems are identified which must be overcome for the system to be both practicable and usable in real time. An algorithm to extract the dynamic systems' equations in various forms is discussed. An overview of the software which has been developed is given, as is a description of the method used to pre-filter data before it is used.

Chapter 5 discusses the methods that I have developed to detect faults in noisy, imprecisely known, dynamic systems. A fault is defined, state estimation described, system model accuracy discussed and fault detection considered.

Chapter 6 examines the methods for diagnosing multiple simultaneous faults, including parameter estimation and how Reiter's [68] and de Kleer's & Williams' [10] [11] idea of conflict sets can be adapted to cope with the imprecise nature of hypotheses based on noisy, inaccurately known systems. Limitations due to the nature of all dynamic systems are then examined and finally a method of determining which hypotheses are the most likely ones is discussed.

Results of tests on a real dynamic system are given in chapter 7. It is shown how false alarms are avoided, faults are detected and single component faults are diagnosed. The method for diagnosing single and multiple faults in the situation when multiple faults are considered possible, is also discussed.

In chapter 8 conclusions are drawn and the results discussed. Implications, limitations and improvements which could be made when using this approach to detect and diagnose faults in real systems, on-line, in real-time are highlighted.

Chapter 2

Literature Survey.

This chapter is divided up into five sections; these relate roughly to the different areas which have been involved during the development of this project. These sections are *Fault Detection, Fault Diagnosis, Modelling, Application Examples* and *Constraint Propagation.*

2.1 Fault Detection

Fault Detection is the method by which the presence of a fault is detected; we are not trying to find the nature of a fault, but rather is there one or isn't there one. As the demands for quality, reliability and safety have increased, so has the need for the improved detection of faults. The cost of a fault going undetected can be immense when consideration is given to the possible waste of the materials used by a process, the delay caused to production, possible damage to the process plant and of course danger to the operators and the environment. A great deal of work has been done over the years, looking at different approaches and methods to improve fault detection techniques. The most popular approach is based on using one or more models of the system [55], these can be either quantitative [21] [27] [41] [48] [76] or qualitative [4]

[16] [17] [18]. Work has also been done using expert/knowledge based systems [15] [65], and the advantages of modelling the system with hierarchies has been looked into [80].

The major objectives of this work have been :-

1. making the detection of faults as accurate as possible, i.e. reducing false alarms to a minimum but at the same time trying to avoid ignoring any genuine faults;

2. reducing to a minimum the time between the actual occurrence of a fault in a system, and its detection by the system monitor;

3. investigating the significance of system noise and model uncertainty on the points 1 and 2 [13] [14].

Points 1 and 2 compete with each other in that, if the detection time is reduced to a minimum, then there is less information about the faulty system upon which to base a decision about the presence of a fault. This means the chances of producing a false alarm are increased, and also the possibilities of missing a genuine fault are heightened. The third point is encountered when a fault detection schema is implemented on a real system. It results in less accurate and delayed detection of faults which counters the objectives of points 1 and 2.

The area of fault detection/diagnosis overlaps with that of model validation in that fault detection assumes the plant is incorrect and model validation assumes the model is incorrect. Both of these areas use a system and a model of that system. They attempt to find out if there are any significant differences (fault detection) and if so, what the differences (fault diagnosis) [29] are.

Details of the development of fault detection can be found in the survey papers by Wilsky [78] and Isermann [41].

2.2 Fault Diagnosis

Fault Diagnosis is concerned with finding what the cause of a fault is, i.e. what has gone wrong and how it has gone wrong. The work in this area can be divided up into a number of sections and each of these is described below.

2.2.1 Fundamental Fault Diagnosis

Important work on the generic diagnosis of faults has been carried out by de Kleer and Williams [10][11], and, Peng and Reggia [66] and a general theory of diagnosis without reference to domain or nature of faults has been developed by Reiter [68]. The work by Davis [7] was also an important step in the development of this theory.

This theory is based on the systems ability take a number of observations [1]. Using these observations of how the system is behaving, a comparison is then made with how the system should be behaving according to the model. Various hypotheses are then tested by simulating faults in the model to find one or more hypotheses which match the observations. This part is dependent upon the domain in which the diagnosis is being undertaken[2]. A set of rules is then followed to find the most likely hypotheses.

Survey papers on the diagnosis of faults specifically in dynamic systems have been written by Frank [19] and Mironovski [59]. Some important practical aspects of fault diagnosis have still to be fully addressed. For example, how to cope with system noise, how to deal with inaccurate system models (a system model can never be 100% accurate) and how to diagnose in real time without using a super computer for the analysis. Diagnosis methods are appearing [62] [1] which are based upon or contain aspects closely related to Reiters' theory of diagnosis. One of the arguments

[1] A measurement taken from all the sensors on the system at the same instant of time is termed as one observation.

[2] The implementation of this for simple logic and electrical circuits is relatively straight forward. For dynamic systems, however, it is much more difficult.

he puts forward is that when additional observations are considered, it is possible to prune the set of possible hypotheses according to system observations which are not consistent with a particular hypothesis. This is true unless the inconsistencies are due to noise on the system measurement or modelling inaccuracies, in which case a correct diagnosis may just have been pruned.

2.2.2 Quantitative methods based on Structure and Function

These methods are based on a mathematical model of the system. The model is in the form of a series of inter-connected subsystems, each one containing a mathematical description of its own behaviour. Each subsystem should, ideally, represent some part of the real system, e.g. a pipe, a tank, a resistor, an axle. The structure of the model is defined by how these components are connected, and the function of the system is the result of the individual component functions, combined with the structure of the system. Here analytical redundancy is used to provide the information needed to locate the fault.

The methods generally consist of either using a model and attempting to determine which areas of the model no longer match the corresponding area of the real system (thereby locating the fault) [7] [8] [75] , or using a number of different models of the system, each one corresponding to a specific fault [37] [27] (in essence, a series of individual fault observers). The observer which matches the systems behaviour is deemed to be the one which describes the fault. There is quite a lot of overlap between these two methods as they are both working on the same underlying principle that most of the system is working, i.e. most of the model of the system is correct, and only part of it is malfunctioning. Model based methods can use some form of parameter estimation [25] to assist in finding which component is faulty and evaluating

a parameter which will describe the new behaviour of the failed component [79] [30] [31].

A very useful summary and introduction to some different methods of fault diagnosis together with application examples and results can be found in [63].

2.2.3 Qualitative methods based on Structure and Function

The models used in these are the same as the ones used in quantitative methods except the description of each subsystem is not described by a mathematical relationship but rather as a qualitative relationship. Real values measured from the system are put into ranges and each range is then given a qualitative value. These qualitative ranges of values are then used throughout the model. For example, if a valve is considered which had a positional sensor giving readings of 0 to 100%, it might be encoded thus as shown in table 2.1.

Range of measured value (%)	Qualitative value
0 - 5	Closed
5 - 25	Just_Open
25-75	Halfway_Open
75 -100	Fully_Open

Table 2.1: Converting real values to ranges.

The behaviour of the valve may then be described by a set of qualitative rules such as :-

$IF\ \ (Valve = Closed)\ \ THEN\ \ Flow = No\_flow$

$IF\ \ (Valve = Fully\_Open\ \ AND\ \ Pressure = High)\ \ THEN\ \ Flow = Large$

$IF\ \ (Valve = Fully\_Open\ \ AND\ \ Pressure = Medium)\ \ THEN\ \ Flow = Medium$

$IF\ \ (Valve = Halfway\_open\ \ AND\ \ Pressure = High)\ \ THEN\ \ Flow = Medium$

etc

These qualitative relations can also be applied to the system states and the rates of change of signals within the system.

Such a system model can be in a number of qualitative states, and when in any one particular state, rules can be generated which determine which states the system might move to next. These rules are derived from the functions of time which describe the systems behaviour. These rules together with temporal qualitative system simulation[74] are making qualitative detection and diagnosis more and more usable in practical situations.

This approach has had success in cases where the speed of a diagnosis is not of great importance,[3] or where the possibility of a multiple fault is significant, i.e. a number of faults occurring simultaneously, and when an accurate evaluation of the size of the fault is not required [12]. That is, the location of the fault is found, but its size cannot always be accurately determined. Qualitative methods may be helpful in the case of multiple faults because of the comparatively large amount of processing required by quantitative methods when considering the many possible combinations of multiple faults. This method, when compared to quantitative methods, is in general, less accurate, but is computationally less demanding and can generally cope with a larger range of systems.

One feature which seems to appear often in qualitative diagnosis systems is the relatively high number of sensors attached to the system being monitored[12] [62]. It would appear that qualitative models/simulations suffer quite badly when the number of components is large and the number of sensors is small, i.e. qualitative signals have to pass through a number of components.

There has also been work done on using both qualitative and quantitative methods to describe the system and model it's behaviour [21].

[3]The diagnosis process can often not begin until significant transients are present.

2.2.4 Expert systems

An area which has recently received attention is the use of expert system/rule based method for diagnosis [18] [61] [65]. Here, a list of rules or facts are entered about how the system behaves when faults occur. When a fault occurs, facts which are proved to be true or false are used to prove or disprove other facts, and a chain effect occurs where the expert system starts deducing the state of the system and what faults are likely to have occurred. The problem with this method is that the rule base must be complete and, as with methods such as fault tree analysis, it can only diagnose a problem which has been specifically represented in the rule base. A further problem is the structuring of the information in the knowledge base; this is often haphazard and quickly becomes confusing and difficult to manage when the system being diagnosed starts to become complicated. These kind of methods can work well with small systems, but can only work with larger system after a great deal of careful work has been done to represent the systems' behaviour in the rule base. Work has been done to incorporate how components and systems function as well as experiential information [15]; this is still heavily system dependent and still requires a large effort to correctly fill the rule base.

2.2.5 Other Approaches

A number of other methods have also been applied to the fault diagnosis area; these include algorithmic approaches [72] [47], neural networks [77], hueristic learning [65] and hierarchical fault diagnosis [36].

2.2.6 Multiple faults

Work has been done in this area at a fundamental level [11] [10] [68], but practical implementations and examples are few. Here, the principle is the same as for a single

fault except that more use is made of redundant information, usually in the form of having more sensors on the system. In dynamic systems, more information can also be gained by considering the dynamic behaviour of components within the system.

2.3 Modelling

An important aspect of all fault detection and diagnosis implementations is how the system is to be represented/modelled. This is important for two reasons. Firstly, the model has to be produced either from already existing documentation or from someone who either knows the system well or has specifically studied the system for the purpose of fault diagnosis. Secondly, the modelling technique used should be capable of containing all the information necessary for fault detection and diagnosis. The appropriate method will depend, to some degree, on the domain in which the fault analysis is to be performed. The representation language used for a digital electronic system and a process plant must model substantially different types of information and behaviours, it would therefore be unlikely that the same language of representation would be used for both of these different domains.

Broadly speaking, there are three main approaches which are being used in fault detection/diagnosis; these are expert systems, qualitative analysis and quantitative analysis. Each of these have different merits which depend upon the size of the system, its complexity (both physically and mathematically) and the sensitivity and accuracy required for the detection and diagnosis of faults. There are four main aspects of models which are commonly used for modelling, these are structure, behaviour, causality and purpose [20]. Qualitative and quantitative analysis are both based upon models of system structure; its' behaviour[4] and often at least some aspects of causality. Expert systems are based more upon behaviour, purpose and again

[4]Qualitative analysis uses a more lose description of behaviour than quantitative analysis.

often some aspects of causality. The choice of which model description is best to use depends upon which diagnosis scheme is employed and also the domain of the system which is studied. Discussions on the issues raised here can be found in [1], [20], [46], [21] and [5].

Alternatively a model of the system may consist of all of the system equations lumped together. This may start off as individual component equations joined together according to the structure of the system, or it may be an equation which approximately matches the behaviour of the system. The latter requires that the parameters in the equation be identified which results in a lumped system description with lumped parameters. Such a model has been effectively used in [2] for diagnosing damage to offshore structures and rotating machines.

The approach adopted here is to use bond graphs as the base representation language. A thorough introduction to bond graphs can be found in [45], [44] and more recent developments in [28].

2.4 Application Examples

Failure detection and diagnosis methodologies have been implemented and tested in many real life situations. For instance, [40] [57] [56] use fault tree analysis to identify faults in machine tools, Leary [51] uses constraint suspension techniques in a laboratory simulation of liquid process equipment, a multiple model/observer method is used in [55] for identifying faults in a experimental heat exchanger, an example applied to the loading of liquid oxygen is given in [70], analytical redundancy applied to aerospace space systems is explored in [64].

2.5 Constraint propagation.

Constraint propagation is one method which can be used for deducing the value of signals which are not measured. These deduced signals can then be used for further deduction. This has been a recently rapid growing area, originating from the artificial intelligence community. Using constraint propagation in this area is basically a formalism for passing signals around a component model of a system. Problems were encountered where loops such as feedback occurred in the model but, for many cases, this has been overcome using simultaneous constraint satisfaction facilities such as those used in the constraint programming language CLP($\Re$)[5] [42] [43] [49] [50] [39]. Development and investigations of constraint propagation/satisfaction/suspension can be found in [38] [3] [54] [6]. Constraint propagation techniques have been applied in many domains where there is a problem to be solved [58] [50] [43]. These ideas have been applied in a theoretical sense to fault diagnosis [36] [8] [7] but it is not until recently that these ideas have been applied to dynamic systems [51] [52] [27].

[5]Constraint Logic Programming in Real Number

Chapter 3

Background Theory and Principles.

In this chapter some of the theory and principles upon which this work is based will be discussed. The limitations and disadvantages in some of the current methods will also be shown, these will be addressed, later, together with possible solutions.

The fundamental principles involved in *constraint propagation* will be looked at in detail and the use of *constraint satisfaction* to assist in detecting the occurrence of a fault and the use of *constraint suspension* techniques to help locate the fault and determine its size (diagnose a fault) will be examined.

Noise on the measurements taken from the system will adversely affect the ability of any diagnosis system. The effects of noise will be looked at briefly here, and it will be noted that the presence of noise cannot be ignored. Finally a brief look will be taken at the implications of considering the possibility that more than one component has failed. This area will be discussed fully in chapter 6, but one should be aware of some of the implications from the outset.

3.1 System models

The fault detection and diagnosis method which is presented here is based on the structure of the system and the function of the components [7] [8]. That is, the parameters of the system are not lumped together to give a compact model of the systems characteristics. The structure (or topology) of the system is represented, i.e. how the different components which make up the system are connected, and the function of each component is explicitly represented. The overall behaviour of the system is then obtained by considering how all the components interact. Use will also be made of the causal relationships between the different components which make up the system.

If it were possible to measure every signal on the real system, then the detection of faults and their diagnosis would be relatively trivial. Since sensors are expensive to purchase, install and maintain, and often the signals of interest cannot be measured, it is necessary to use models of systems and analyse these using just a few measurements from the real system. These measurements are then extrapolated through the model of the system and used to try to determine the value of signals that cannot be measured. By looking at these extrapolated signals, and the expected values of the signals, given the system inputs, one can hopefully detect the presence of a fault and then find its location and size.

The way the system is modelled is very important. It is necessary to break the system down into components or subsystems (function), and also clearly define how these are connected together (structure). As an example, consider figure 3.1, which shows an electrical resistor and capacitor in series with a voltage source. Here, the way in which the wires connect the components is the structure, and the way in which the components behave is the function; together they represent the system. Table 3.1 shows these separate parts which make up the system. It is possible to break up

virtually all physical systems into component subsystems and to define a structure which represents the interconnection of the components. For example a fluid system could be broken up into tanks and valves connected by pipes, or a mechanical system into springs and masses connected by links.

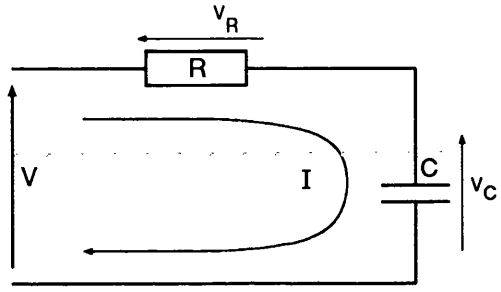


Figure 3.1: A schematic of an R-C circuit.

Structure	Function
$I_R = I_C = I_{source}$	$V_R = I_R \times R$
$V + V_R + V_C = 0$	$V_C = \frac{1}{C} \times \int I_C dt$

Table 3.1: The structure and function of the R-C circuit.

Many systems are some combination of mechanical, electrical, hydraulic or thermodynamic systems. To enable these multi-domain systems to be handled in a uniform way, bond graphs have been utilised to provide a method of modelling them [45] [32]. This method is versatile, powerful and easy to use. It is certainly not possible to give full details of bond graphs here, but some of their features will be discussed later to help clarify the fault analysis procedures.

3.2 Bond Graphs

The bond graph for the R-C circuit in figure 3.1 is shown in figure 3.2. Note that the structural part has a different representation, but the components can still be seen. This is because a bond graph represents the flow of energy through the system, each bond carries energy from a component at one end to the component at the other end,

in the direction of the half arrow —. As a further example, a bond graph of a dc motor is shown in figure 3.3; this also gives an indication of the different parts of the system. A dc motor converts electrical energy into mechanical energy and it therefore covers two domains, but the bond graph technique allows these different domains to be modelled in the same fashion and therefore relieves us of the burden of having to interpret models from different domains in different ways.

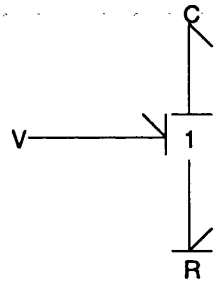


Figure 3.2: A bond graph of an R-C circuit.

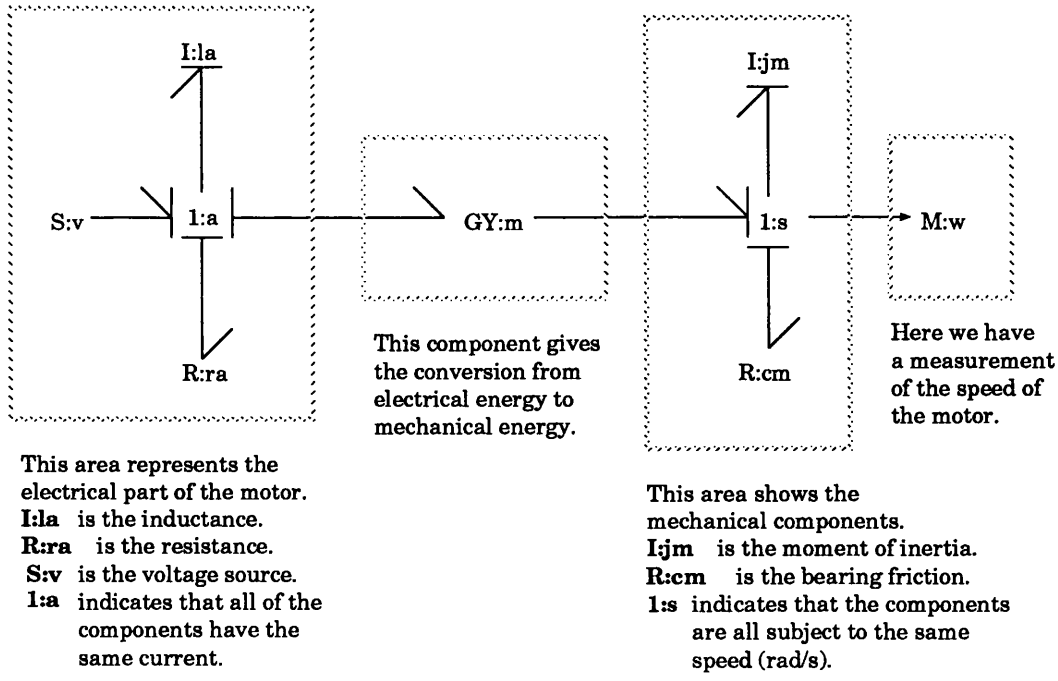


Figure 3.3: A bond graph of a DC motor.

Another aspect of bond graphs is their easy extensibility. Consider a system of a dc motor driving a pulley which lifts a weight. All that is needed is to get a bond

graph for a pulley and weight and simply connect it with a bond to the rotor part of the motor. This is in marked contrast to the block diagram way of doing things, which would need a completely new analysis of the system and re-drawing of the system model. There are a number of tools which have been developed to manipulate bond graphs [28]. These take a drawing of a bond graph as input and are then able to analyse the system in a number of different ways.

3.2.1 Signals

Each bond on the bond graph represents the flow of energy in the direction indicated by the half arrow. Each bond has two variables associated with it, one of type effort, and one of type flow. The nature of the effort and flow variables changes depending upon which domain they are in, but in all cases, the product of the effort and flow variables is power. In the electrical domain, the effort variable is voltage and the flow is current, their product being power. Below, table3.2 shows the effort and flow variables for some other domains.

Domain	Effort	Flow
Electrical	Voltage	Current
Mechanics	Force	Velocity
Hydraulics	Pressure	Volumetric flow
Thermodynamics	Temperature	Entropy flow

Table 3.2: Some effort and flow variables for different domains.

These signals can be thought of as the inputs and outputs of components. For example with an electrical resistor, it will be connected to a bond and the bond will be associated with a voltage and a current. In this case the voltage is the voltage across the resistor and the current is the current flowing through it. The product of these two is the power which the resistor is dissipating. If I is know then it is possible to say $V = I.R$, conversely if V is known then $I = \frac{V}{R}$. Exactly which of these to use is decided by assigning causality to the systems components i.e. which components

cause signals to change and which components are affected by this, causality is too complex an issue to go into here, but it is possible to assign causality automatically. Causality is represented on a bond graph by the short perpendicular strokes at the end of bonds.

3.2.2 Components

There are four basic types of components which are used to construct bond graph models. These are Sources, Dissipators, Stores and Transfer components. Each of these will now be discussed in relation to fault analysis.

Sources. These components source either the effort or the flow variable on the bond that they are connected to. They also sink the co-variable which isn't sourced. An example may be an ideal battery, a fixed voltage appears across the battery and it will provide a varying current depending on the load that it is connected to. These are used to represent system inputs, and are not strictly part of the system, but represent an interface with the outside world. For this reason, detecting faults that arise from these components will not be looked at. If it is necessary to detect and diagnose faults in the interfaces of the system, then these also have to be explicitly modelled as part of the system.

Dissipators. These components absorb energy from the system. In their simplest form the component consists of one parameter which is a constant of proportionality between the effort and flow variables on the bond that it is connected to. An example of this is an electrical resistor, $V = I.R$ where R is the constant. Of course, the relationship need not be linear; for instance the flow through a pipe, $Flow = C_D\sqrt{Pressure\ difference}$, can also be represented by a dissipator. A fault in this kind of component can be modelled as a change in the component's parameter; e.g. if the resistor burns out, gets broken or gets shorted, the result

is effectively a change in its resistance, as seen by the rest of the system. Detecting and diagnosing faults of this nature in these components will be looked at.

Stores. These components accumulate either the effort or flow variable. An example of a flow store is a capacitor, it stores current. An inductor on the other hand stores voltage and is an example of an effort store[1]. These represent the states of the system. They have a parameter associated with them which represents their physical coefficient, eg capacitance; inductance; for a water tank, cross sectional area; for a rotating mass, its moment of inertia. Faults in these components can be detected and diagnosed in a conceptually similar way to dissipators, except that the value of the state has also to be estimated. This will be discussed later.

Transfer components. These are usually used to convert energy from one domain into another. They are connected to two bonds, one from either domain. They do not store or dissipate energy, they just convert it into a different domain. They usually have one parameter associated with them which is a proportional constant, but they contain two relations. An example of this appeared in the diagram of a dc motor earlier. Here component GY converts the current into a torque ($Torque = M \times Current$), and also converts the velocity of the rotor into a back emf ($Emf = M \times Velocity$), where M is the parameter of the component. Again, faults in these components are found in a similar way to dissipators, except that there are two relations which depend on the same parameter. This will be discussed later.

These are the basic components needed to model a system, the actual relationship that any particular component contains is not constrained from a bond graph theory

[1]By convention an effort store is represented with an 'I' and a flow store is represented with a 'C' on a bond graph.

point of view, but the more complex the relation is, the harder it is to analyse. The condition that all component relations must be linear is not imposed. However, if the relationships are too complex then the system equations cannot be solved in the form that is required, and therefore it will not be possible to carry out as full a diagnosis as would have been preferred. It should be noted however that this situation can be helped by using additional sensors on the system, i.e. having more information about the internal behaviour of the system.

3.2.3 Structure

The structure of the system is represented by the bonds which join components together, and by the '0' and '1' junctions. These junctions define whether components are connected in parallel or in series. Each junction contains two constraints.

For the '0' junction, the constraints are:-

1. All of the effort variables on the bonds connected to it must be equal.
2. All of the flow variable on the bond connected to it must sum to zero.

For the '1' junction, the constraints are:-

1. All of the flow variables on the bonds connected to it must be equal.
2. All of the effort variable on the bond connected to it must sum to zero.

Looking back to figure 3.2, a '1' junction can be seen. In the electrical domain, the flow variable is current and the effort variable if voltage. This junction states that the current (flow) on each component is the same and the sum of the voltages is zero, i.e. the components are in series. If a '0' junction was used, then the result would be a voltage source in parallel with a resistor which was in parallel with a capacitor.

3.3 Constraint Propagation and Suspension.

Constraint propagation techniques can be used with any system which can be represented as a number of inter-connected components, and has typically been used for the analysis of electrical circuits. The possible uses of constraint propagation are, however, much wider than this, it could also be used to look at mechanical systems, economic systems, qualitative system descriptions of industrial processes or models of social systems.

Firstly the basic concept of constraint propagation and suspension is introduced, two examples will then be used to demonstrate how these techniques can be used for simple simulation and fault analysis of non-noisy, non-dynamic systems.

3.4 Constraint Propagation and Fault Detection

3.4.1 Constraint Propagation

Constraint propagation is the term used to describe the method by which the values of signals in a model are found, which are not measured from the system. So, for example using a simple one input, one output system, one may wish to measure the input to the system and then propagate the effect of this value through a model of the system and predict what the output should be. Consider the simple system in figure 3.4.

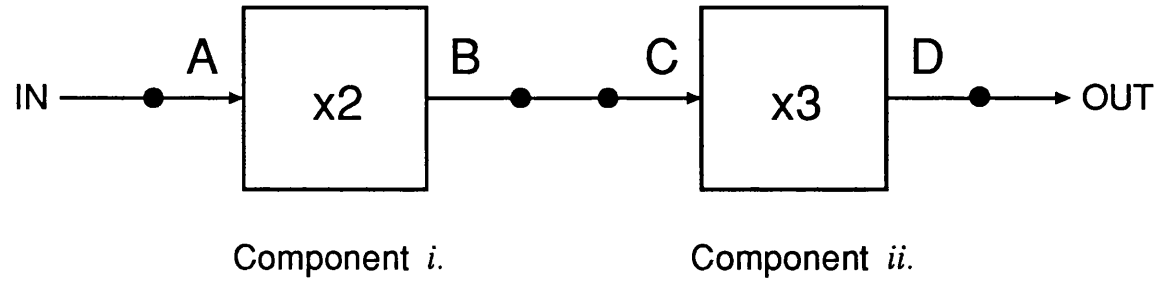


Figure 3.4: A simple 1-input 1-output system.

The system consists of two components *i* and *ii* which are both multiplying components, but they have different parameters, namely 2 and 3 respectively. The value of the input, A is multiplied by 2 giving B, which is then multiplied by 3 to give D, the output. This demonstrates in a simplistic way what constraint propagation is. The two components are constraints, and in addition, the input to component *i* is constrained to have the same value as the input to the system, the input to component *ii* is constrained to take the same value as the output from component *i* and finally, the output from the system is constrained to have the same value as the output from component *ii*.

Propagation is carried out as follows. Firstly the input to the system has been measured, suppose it has the value 4. Now look through the constraints to see what this tells us about the other values of signals in the system. All that can be done at this time is say that A is equal to 4. The process is then repeated using this new information leading to B is equal to 8, by using component *i* as the constraint, this leads to C also taking the value of 8, by the constraint '$B = C$'. Next, the component *ii* is used as the constraint to give D equal to 24, and finally the output takes the value 24.

This may seem a long winded description, but it is necessary to be fully aware of all the steps that are carried in propagating signals through the constraints of the system.

Other examples of similar system analysis can be found in [8], [36], [50] and [39].

3.4.2 Fault Detection

Suppose, in the example above, that the value of the output from the system was also measured. Comparing this measured value, with the value just obtained from propagating the measured value of the input through the model, gives us a way of

determining whether or not the system is behaving correctly. (If the values are the same, the system is functioning correctly, otherwise something has gone wrong). This is one way in which a fault can be detected.

Alternatively, the measured value of the system output could be taken and propagated backwards through the model to arrive at a value of what the input should be, for the measured output to be correct. This could then be compared with the measured input. Again, if they are the same there is no fault, if they are different, then there is a fault.

As a final alternative, the input and the output could be measured. The next step is then to propagate the input forwards through the model and at the same time propagate the output backwards through the model. This way there will be two ways of calculating one of the signals inside the system, for example, $C = 2 \times input$, but also $C = output \div 3$. Again, if these are equal there is no fault otherwise there is a fault.

This is a simple method for simple systems of finding out whether the system is behaving correctly and can be summarised as follows.

If all of the constraints of the system can be satisfied, then the system is functioning correctly and there is no fault. This was demonstrated for simple, non-noisy systems by Gawthrop and Leary [52],[51] using constraint propagation techniques, others have also found this to be true, although without explicate reference to constraint propagation. These include Davis *et al* [8], Genesereth [36] and Yung and Clarke [80].

3.5 Constraint Suspension and Fault Diagnosis

3.5.1 Constraint Suspension

Constraint suspension is a method of selectively relaxing constraints within a system. It is a useful method in fault diagnosis, used after a fault has been detected. This is now explained.

If it is possible to determine all of the signals within a system with one or more components suspended, then it will be possible to deduce the behaviour of the suspended components without reference to their previously assumed behaviour.

This new behaviour is consistent with this observation of the system. If the deduced behaviour of the components are different from their defined behaviour, then one may say that the fault within the system can be explained by these components changing their behaviour.

An observation of a system is a set of measurements taken from the system at one instance of time. If the behaviour of the component has been deduced from just one observation, then further observations are needed which corroborates the deduced behaviour from the first observation. Then one can say with confidence that a fault in these components will explain the fault in the system, although it must be noted that this may not be the only explanation of the fault. Because of the nature of system structures and measurement locations on a particular system, it is often possible to arrive at more than one valid explanation of an observed fault, as will be shown shortly.

3.5.2 Fault Diagnosis

To diagnose faults, and in particular to deal with the possibility that more than one component has simultaneously failed, advantage will be taken of the work done by

Reiter with his theory of diagnosis [68]. To do this it is necessary to ascertain whether one or more components, when considered to be faulty, would make the rest of the components appear to be behaving correctly. This will now be demonstrated for simple systems, and this topic will be returned to later with more 'realistic' systems.

Consider the example from the previous section, shown again in figure 3.5. If the input was measured as having the value 4, and the output was measured having value 32, then this does not agree with the expected behaviour of the system so one can say that there is a fault in the system. From the input, propagation can take place through the constraints to C. C can then be given the value of 8. Similarly propagation can take place from the output to D. D then equals 32. Looking now only at component *ii*, the situation is as shown in figure 3.6. It can be seen that the values of 8 and 32 can be made consistent by giving component *ii*, the relation ×4.

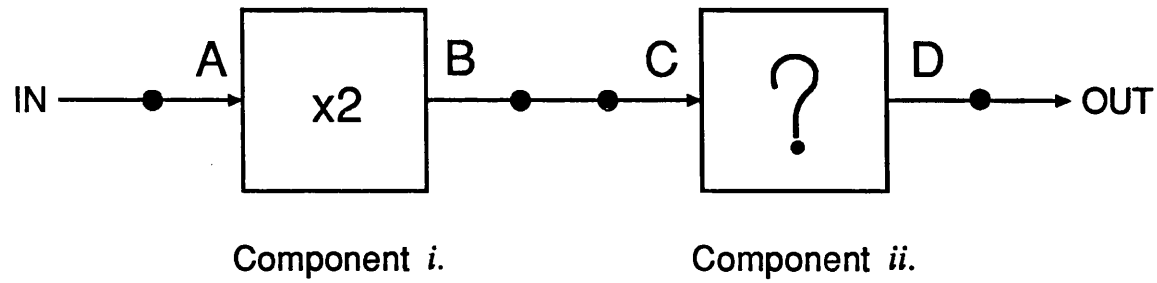


Figure 3.5: A simple 1-input 1-output system, Component *ii* is suspended.

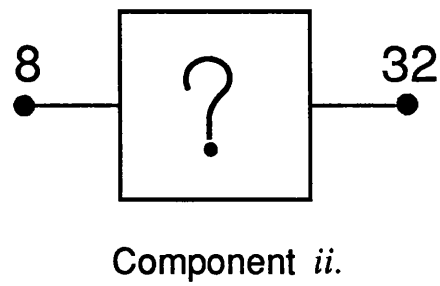


Figure 3.6: Component *ii*.

This assumes of course that component *ii* is behaving as a multiplying component, that is, it is still behaving in the same manner as before, but the value of its parameter has changed from 3 to 4. In general, this will be the case for the type of systems that

are concentrated upon here; the failure of a component will be assumed to result in only the components parameter changing. For example, if a resistor fails, it would be very surprising if it suddenly started behaving as a capacitor.

In figure 3.5 it would also be possible to suspend component *i*, propagate signals to either side of it and say that the fault can also be described by the parameter of component *i* changing from 2 to $2\frac{2}{3}$. In this case, there are two solutions to the fault observed in the system; the parameter of either component *i* or *ii* has changed and in this case it is impossible to differentiate between these two possibilities and so both are valid diagnoses of the fault.

It is also possible for both of the components to be faulty, for example if the parameter of component *i* had the value 1, and that of component *ii* had the value 8, the observed measurements would again be obtained. It is always possible to expand a fault hypothesis in this way but it does not help us in the quest for the true fault unless more information is available which can be used to verified this. If the value of either B or C could be measured, more information would be present and it would be possible to find out which component was the faulty one, or indeed if both were faulty.

The reasoning method for multiple fault combinations is described later, but the basic principle for how a fault is diagnosed should be clear. In summary, a component is suspended (i.e. do not use its constitutive relationship), and all the other signals in the system are found. This will give values on either side of the suspended component, and from these values it is possible to find a value for the components parameter which makes the signals consistent. If this is repeated with a different set of data, and the same value of the components' parameter is produced, then it is safe to say that a fault in this component would explain the fault in the system.

This is a simplistic way of testing a diagnosis and leaves many questions

unanswered. These will be addressed further on, but it is important to understand the basic methodology before moving on to discuss some of the more complex issues which arise.

In some applications, it is enough to simply show that when a component is suspended, and all of the remaining constraints are satisfied, then that component can be considered to be the cause of the fault in the system both in single fault analysis [8], [35], [36] and in multiple simultaneous fault conditions [11]. However, with noisy, dynamic systems, using quantitative analysis with a limited number of sensors on the system means that, not only is it necessary to show that rest of the constraints are consistent, but also that the estimated parameters for suspended component(s) are constant, before the cause of a fault can be attributed to any particular component. This will be apparent later, but first some of the issues which are common to all types of systems will be examined.

In the following sections, some of these issues will be introduced with examples which will demonstrate more about constraint propagation/suspension. Some of the problems which occur when considering more complex and more 'realistic' systems will be pointed out.

3.6 Binary systems.

This is an example using a circuit for a full adder which Reiter used to demonstrate his theory of diagnosis [68]. The inputs and outputs are allowed to adopt one of two values, either 0 or 1. It will be used here to demonstrate some of the logical considerations of fault diagnosis. The circuit is shown in figure 3.7.

A full adder is used for adding two single binary digits and a carry digit, the result is one digit for the result of the addition, plus a carry digit if the result was more than 1. X1 and X2 are Exclusive-OR gates, A1 and A2 are AND gates, and O1 is

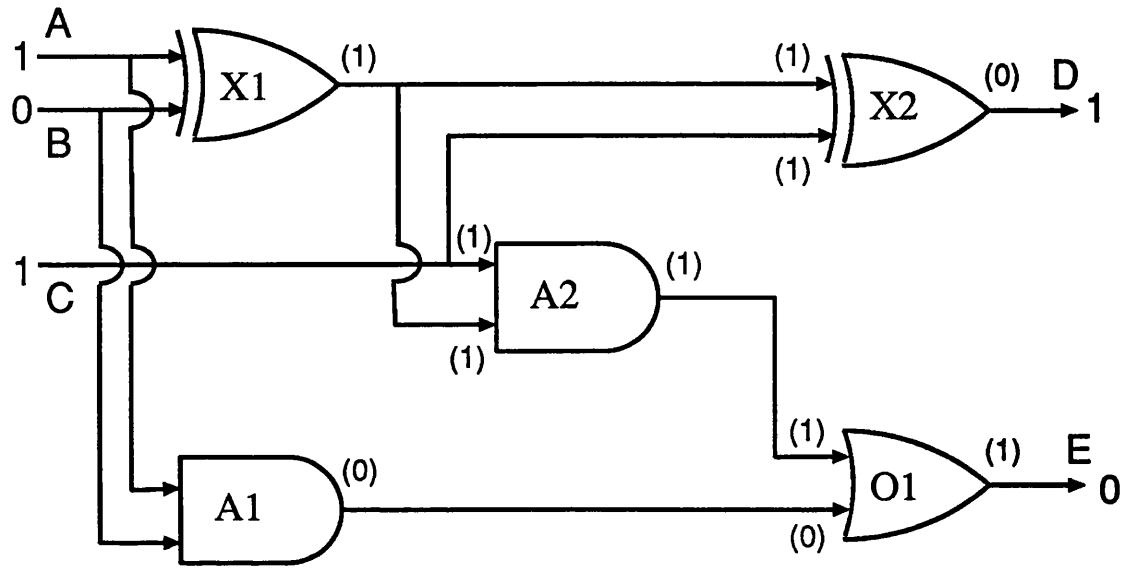


Figure 3.7: Example using a logical full adder.

an OR gate. A, B and C are the inputs to the adder (C is the carry), D and E are the outputs (E is the output carry). The numbers in brackets are the values that the outputs from the gates should be if the circuit is working correctly and is subjected to the inputs shown. The values not in brackets are measured values. As can be seen, the circuit is at fault, but what is the fault?

Using the constraint propagation/suspension techniques described above, this circuit will be analysed to highlight some of the logical aspects of fault diagnosis and some short comings of constraint propagation/suspension as described so far.

Firstly, let us suspend a component and see what happens. Figure 3.8 shows the circuit with component O1 suspended. The arrow heads on the lines connecting the components shows the direction in which signals are propagated. It is assumed that signals do arrive on the inputs and outputs of the suspended component so these could be examined to see if a new behaviour for O1 can be determined which explains the observations. Before this is done though, all of the signals need to be checked for consistency. Looking at the output of X1, it can be seen that if the signals from its inputs are propagated to its output, the value 1 is obtained. Propagating the

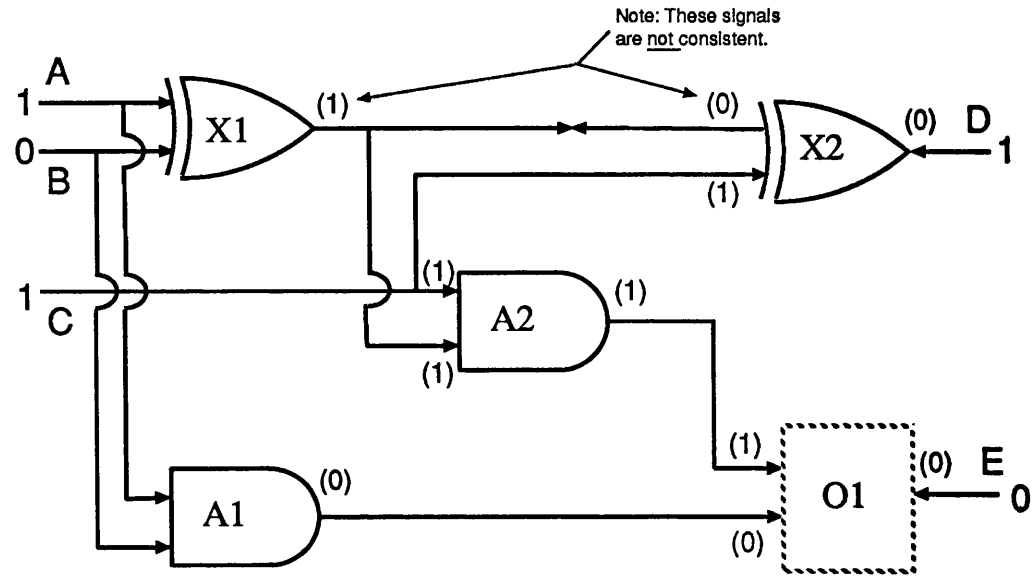


Figure 3.8: A Full Adder with O1 suspended.

measurements from C and D going through X2, the value 0 is obtained for its other input. Clearly 1 does not equal 0, so the signals are not consistent. This means that a fault in O1 alone cannot explain the observations no-matter how it is behaving!

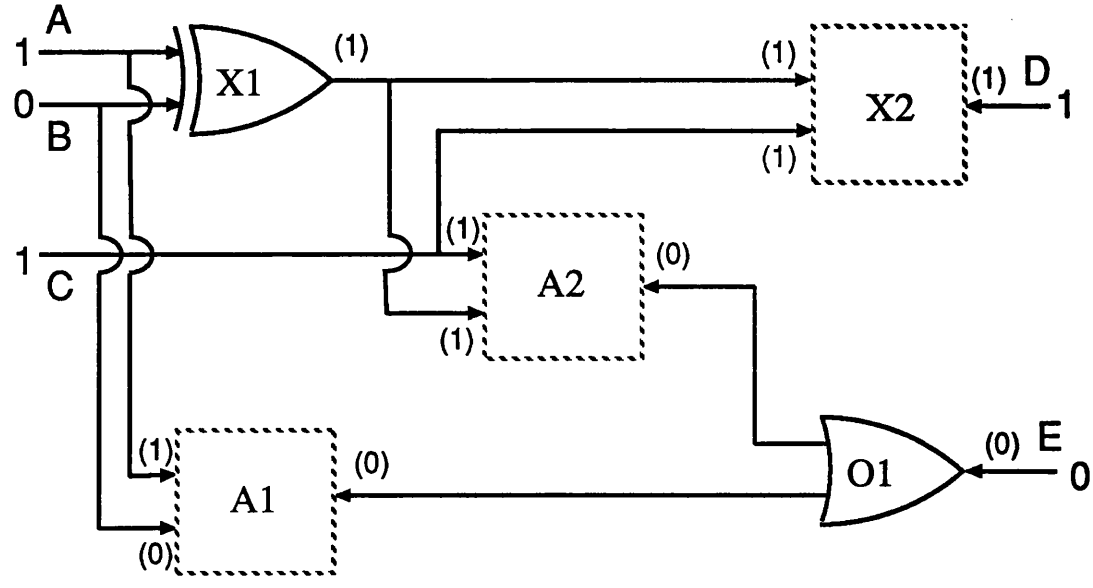


Figure 3.9: A Full Adder with X2, A2 and A1 suspended.

Now let us look at another example, this time three components will be suspended simultaneously. X2, A2 and A1 will be suspended. This is called, at this stage, a

multiple fault hypothesis, if it can be shown that all these three components are faulty this will then be a multiple fault diagnosis. Hypotheses and diagnoses will be represented as a list of components in braces. In this case the hypothesis considered was {X2, A2, A1}.

This situation is shown in figure 3.9. Signals can be propagated all around the system without conflict and therefore this observation {X2, A2, A1} is a diagnosis. The question that arises though is "Is this going too far?" If all the components were suspended one would also get a correct diagnosis. As was pointed out earlier it is always possible to expand a correct diagnosis and obtain another correct diagnosis which is basically a superset of the first one. Is the diagnosis {X2, A2, A1} an expansion of a smaller diagnosis? Note also the two inputs to component O1, these are both zero because of the nature of the OR gate and its output is known to be zero. If the output was 1, it would not have been possible to know if the inputs were (1,1), (1,0) or (0,1). This does not affect making the diagnosis in this case but it could affect identifying the new behaviours of the components which were considered to be faulty.

Figure 3.10 shows the full adder with X2 and A2 suspended. The hypothesis is {X2, A2}. This can also be seen to be consistent. {X2, A2} is a subset of {X2, A2, A1}, therefore {X2, A2, A1} should be discarded and {X2, A2} used instead as a correct diagnosis.

Finally figure 3.11 shows the full adder with X1 suspended, this is also a correct diagnosis but is not a subset of {X2, A2, A1}. There are now two equally correct diagnoses and there is no way to differentiate between them with the information obtained from this one observation. There is a third diagnosis which is also correct, {X2, O2}. There are in total three diagnoses which are all correct and are not supersets of any other diagnoses.

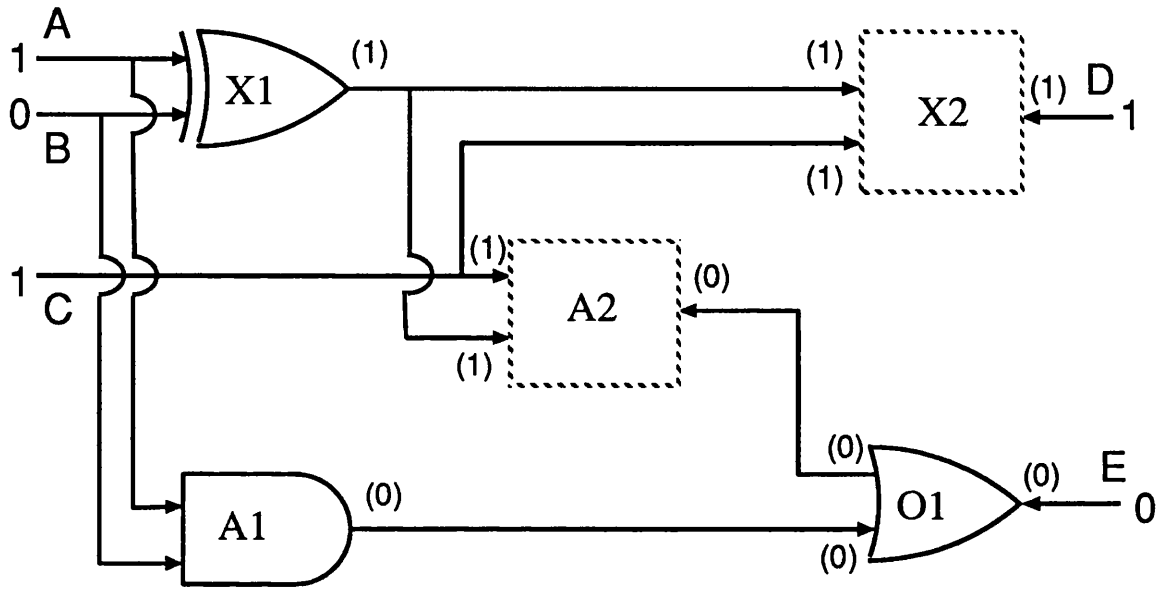


Figure 3.10: A Full Adder with X2 and A2 suspended.

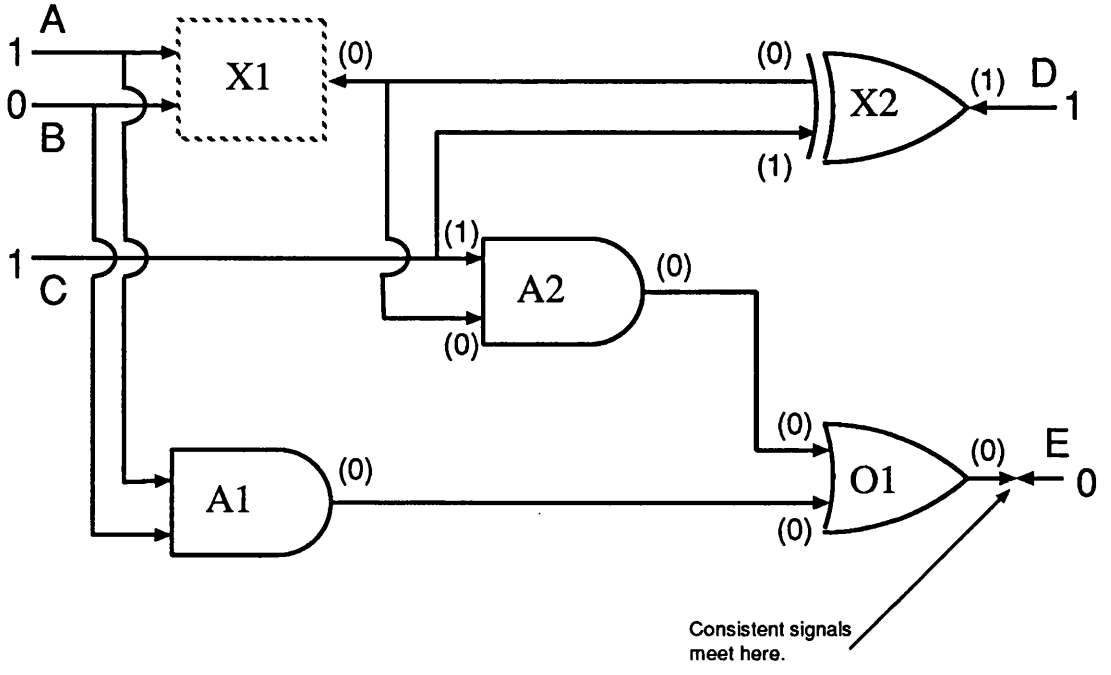


Figure 3.11: A Full Adder with X1 suspended.

There are a number of points that should be noted from the above example.

1. A diagnosis is only minimal if no proper subset of it is also a diagnosis.

2. It is possible to get more than one diagnosis which is a correct diagnosis.

3. When the possibility of multiple faults is allowed, there are many hypotheses which need to be handled.

4. A haphazard approach to testing hypotheses is clearly not practicable when considering larger systems and real-time diagnosis.

Here, some of the fundamental aspects of fault diagnosis have been demonstrated. The four points highlighted above are common to systems in all domains, however it is rarely as straight forward as in a system which deals only with binary numbers. When dealing with real numbers it is not always simple or indeed possible to propagate signals backward through components. If the system contains feedback loops then the ability to solve constraints simultaneously is required, and if the signals contain noise then the signals will never be exactly consistent and other, more involved methods to check hypotheses will be needed. In the remainder of this chapter, the implications of using real numbers will be considered, a brief look at non-linear components will be taken and then the effects of noise and how to cope with it will be explored. An introduction to reasoning about multiple faults will also be given. Noise, non-linear components and dynamic components (components whose output is a function of past inputs) will be examined in more detail in the next chapter.

3.7 Non-Dynamic systems.

Here, two examples will be given to demonstrate some aspects and problems when considering systems which use real numbers. Firstly it will be shown how more

observations of a faulty system can be used to narrow the number of correct hypotheses, and how estimates of component parameters are used to do this. Secondly, it will be shown with a similar example how more observations can also confuse matters and how the introduction of some types of non-linear components can increase the number of hypotheses as well as increasing the complexity of computation. Similar examples of non-dynamic noise-free systems can be found in [10] [11] [8] [36].

From here onwards, components which do not have any parameters will be regarded as part of the structure of the system rather than as a component in which a fault could develop. This is because, if this component were to fail then its constitutive relationship would be fundamentally different, and the component could behave in a whole host of undetermined ways. This would leave us with the problem of trying to match relationships and parameters to the fault and this is equivalent to identifying a fault in the system structure which is much more difficult.

However, it is possible to identify limited failures in the system structure. This can be done by adding additional components to the system with parameters that prevent it from having any influence during normal running but allow for the possibility of a fault developing later. For example, if the system is an electrical circuit, and it is desired that the possibility of a short between two wires should be included, then it can be done by connecting a resistor component between the two wires and setting its resistance to a very high value. If a short then develops, it can be identified by representing it as a change in the value of the resistors parameter from very high to zero. Alternatively, if one wished to model the possibility of a hole developing in a pipe containing fluid, then this can be modelled as a Tee junction where the pipe coming off consists of an open ended pipe to the surroundings with a diameter of zero. If a leak does occur then this can be represented as a change in the parameter

from zero to some positive value which reflects the size of the hole.

Components with parameters will be regarded as having fixed functions, so that a fault in the component will manifests itself as a change in the value of its parameter, i.e. a component that multiplies by a constant will always multiply by a constant, but a fault in the component will cause the value of the constant to change. After a fault occurs it will be assumed that the nature of the fault remains constant, i.e. the new value of the parameters in any faulty components do not fluctuate during the diagnosis process. Components which have multiple parameters will be modelled as a subsystem which contains a number of single parameter components.

3.7.1 Example 1. A System with Linear Components

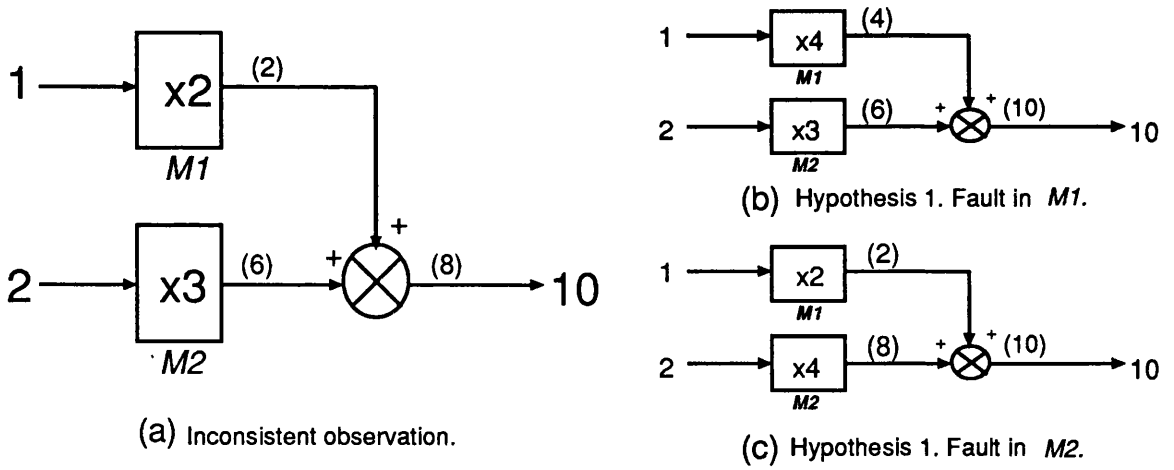


Figure 3.12: A two component system, the first observation.

Here a system which contains two multiplication components, $M1$ and $M2$, is considered. Each of these multiplies its input by the value of its parameters and puts the result on its output. Before a fault occurs, $M1$ has the parameter 2, and $M2$ has the parameter 3. In figure 3.12(a) the system with values for inputs and outputs which have been measured is shown. It can be seen that there is a fault in

the system as the output predicted from the systems model is 8, and the measured value for the output is 10. There are two components, and in this system a fault in either of them would explain the fault. Either the parameter of $M1$ could have changed from 2 to 4, (figure 3.12(b)) or the parameter of $M2$ could have changed from 3 to 4 (figure 3.12(c)). Both of these will explain the observed fault, and there is no way to determine which is the real fault without further information.

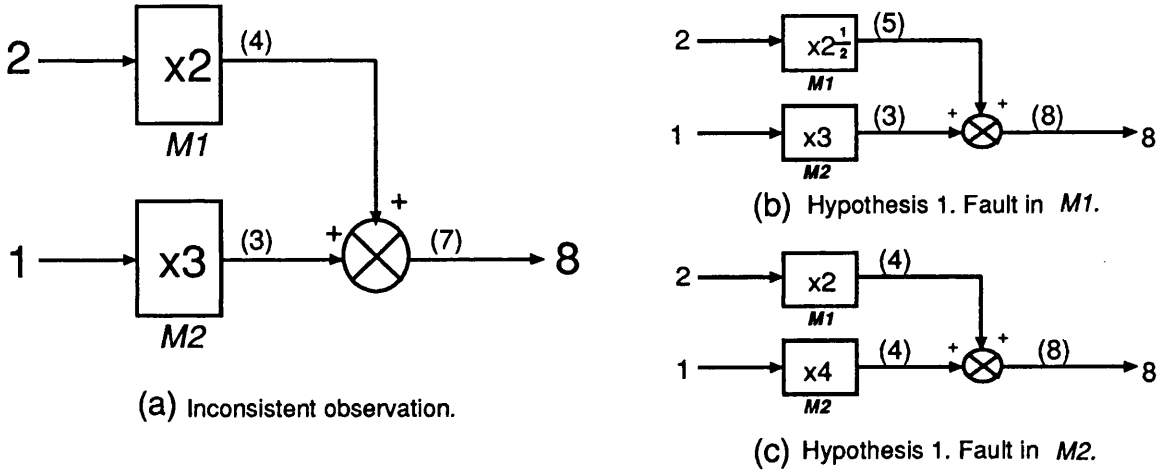


Figure 3.13: A two component system, the second observation.

In figure 3.13(a), the same system is shown, but the inputs have been changed and the new value of the output is measured. This observation can again be explained by a fault in either of the components. Either the value of $M1$'s parameter has changed from 2 to $2\frac{1}{2}$ (figure 3.13(b)) or $M2$'s parameter has changed from 3 to 4 (figure 3.13(c)). Again, there is no way to tell which one is really faulty. These two observations are shown in table 3.3 and when considered together they reveal more information. When both observations are considered, it can be seen that they are both explained by a change in the value of $M2$'s parameter from 3 to 4. That is the behaviour from component $M2$ is the same for both observations. It is therefore reasonable to assume that $M2$ is faulty and has adopted a new value for its parameter.

Observation Number	Hypothesis 1 Fault in $M1$ (new parameter value)	Hypothesis 2 Fault in $M2$ (new parameter value)
1	4	4
2	$2\frac{1}{2}$	4

Table 3.3: Comparing the two observations.

The following points should be noted.

1. Using more observations usually enables us to differentiate between different hypotheses.

2. It is necessary to calculate the value of a component's parameter in order to check that the component's behaviour remains constant.

3. A hypothesis is correct only if it is a valid diagnosis for every observation of the faulty system.

3.7.2 Example 2. A System with Non-Linear Components.

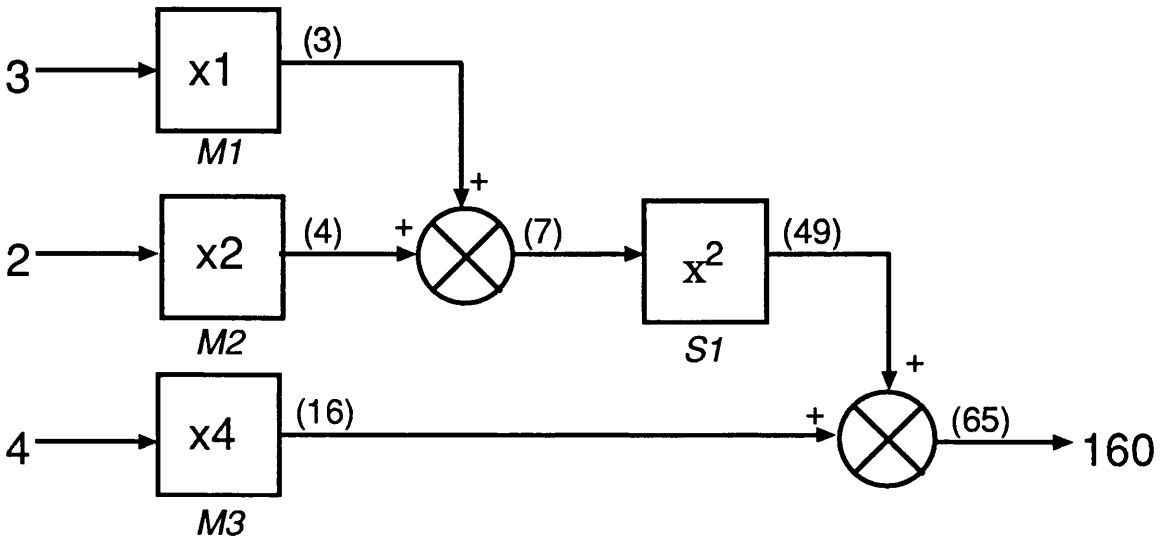


Figure 3.14: An inconsistent observation of the system.

Here the system in figure 3.14 will be examined. This is similar to the first example, except now there are three multiplying components and a square function in component $S1$, $output = input^2$. As can be seen the measured values are not consistent, the measured output is 160 and the expected output is 65. In figure 3.15(a) one diagnosis is shown, $M1 = 2\frac{2}{3}$, figure 3.15(b) shows another diagnosis, $M1 = -5\frac{1}{3}$. There are two correct diagnoses for this observation each of which involve the same component, but with different parameter values. This is due to propagating signals backwards through $S1$, as $12^2 = 144$ and $(-12)^2 = 144$. This kind of effect is very difficult to deal with in practice, especially when the interactions between the components becomes more complicated and the signals contain noise. A list of all the single fault diagnoses, which are correct from this one observation, is shown in table 3.4.

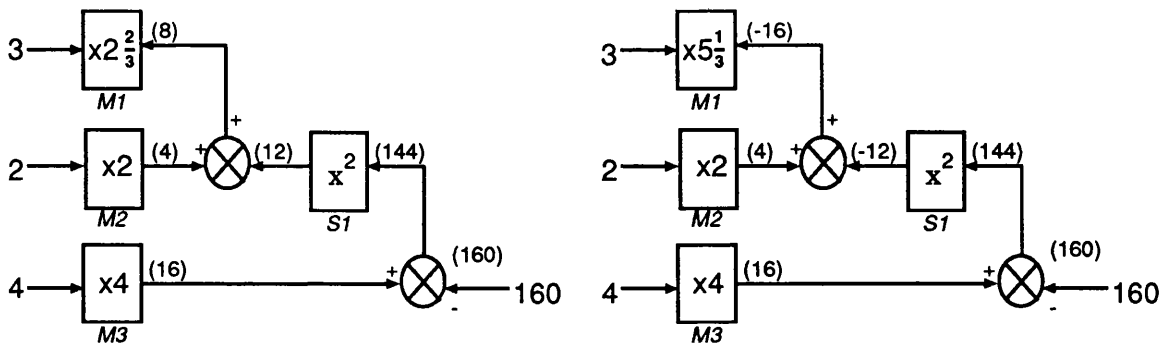


Figure 3.15: Two possible diagnoses of the fault.

A second observation is shown in figure 3.16 and a table for all the diagnoses of this observation is shown in table 3.5. If these results are examined and compared with table 3.4, it is seen that no diagnosis remains the same. All of these diagnoses have been eliminated as being the cause of the fault. There are now two choices,

Diagnosis	Component	New parameter value
1	$M1$	$2\frac{2}{3}$
2	$M1$	$-5\frac{1}{3}$
3	$M2$	$4\frac{1}{2}$
4	$M2$	$-7\frac{1}{2}$
5	$M3$	$27\frac{3}{4}$

Table 3.4: All the diagnoses from the first observation.

either give up with the assumption that the fault cannot be identified or take a closer look to see what is happening. It is not always possible to identify every fault in the system, so the first option is a valid one to make at some stage, but it is too early to take it in this case.

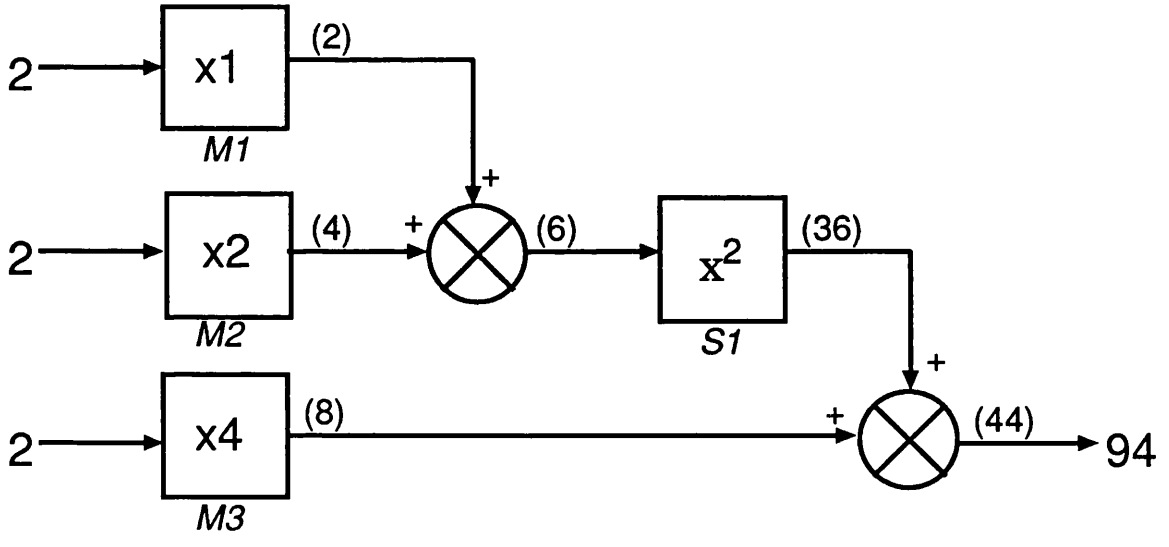


Figure 3.16: A second inconsistent observation of the system.

Diagnosis	Component	New parameter value
1	$M1$	2.6368
2	$M1$	-6.6368
3	$M2$	3.6368
4	$M2$	-5.6368
5	$M3$	29

Table 3.5: All the diagnoses from the second observation.

What these two observations tell us is that the fault cannot be explained by

a parameter in one of the components changing value. Either the parameter of a component is fluctuating, some unmodeled part of the structure has failed or more than one component has failed. In this case the latter is true and the correct diagnosis is displayed in figure 3.17, both $M1$ and $M3$ have failed, the new parameter of $M1$ is 2, and the new parameter of $M3$ is 15.

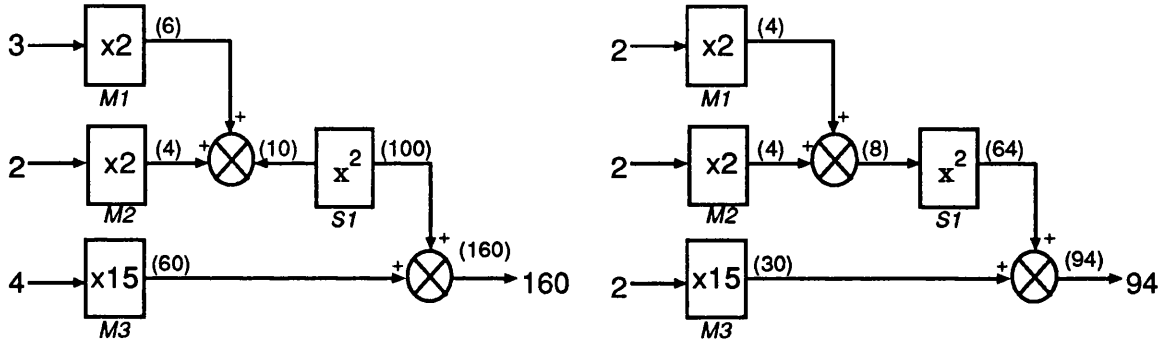


Figure 3.17: The two observations and the correct multiple component diagnosis of the fault.

The points to note are:-

1. components that either give the same output for different inputs or vice versa can greatly increase the number of hypotheses and can be difficult to deal with in complex noisy systems;

2. as wide a range of faults as possible needs to be considered from the start of the diagnosis process. In the example just single fault hypotheses were initially considered, when all of these failed to explain the observations it was necessary to go back and consider multiple faults. It would be more efficient to be able to test the viability of multiple fault hypotheses from the outset;

3. to test a multiple fault hypothesis it is necessary to identify all the parameters of the components in the hypothesis simultaneously. This means that a number of observations need to be considered simultaneously in order to have enough information to find the parameters simultaneously.

When dealing with some non-linear components, it can sometimes be useful to regard one physical component as more than one component in the model of the system. Each of these components will now also have a condition associated with them. This can be a great help in easing the complexity of the mathematics. In addition to this one is also made aware from the outset that these components have different behavioural properties e.g. the behaviour of the component, as well as its output, depend upon the value of its input. This arises, for example, if the static friction component on a moving object was being modelled, the value of the static frictional force depends upon the direction of the velocity of the object, not the size of the velocity. e.g.

1. Static Friction $=C_1$, if Velocity > 0.

2. Static Friction $=C_2$, if Velocity < 0.

3. $C_2 <$ Static Friction $< C_1$, if Velocity $= 0$.

The component x^2 from the previous examples may be implemented as the two components :-

1. $output = input^2$, where $input \geq 0$.

2. $output = (-input)^2$, where $input < 0$.

3.8 Measurement Noise

If signals are being measured from a real system, the measurements will always be subject to some level of error (due to the finite accuracy of the measuring device). Also the system that is being measured will have some degree of noise on the quantities that are are being measured (this may be ripples on the top of a water tank where the level is being measured, electrical noise in a circuit or some small vibrations in a mechanical system). If a fault detection and diagnosis method is to work in practice it is necessary to take account of noise. In the literature this area has received very little attention with regards to constraint propagation, e.g. how to propagate noisy signals.

To demonstrate why this is important for real systems, consider the simple system illustrated in section 3.7.1. You will recall that two observations of this system were taken, the first observation had input measurements of 1 and 2, and an output measurement of 10. The second observation was input measurements of 2 and 1, and an output measurement of 8. Now let us redo these two observations except with the addition of noise on the measurements. The results of this are shown in table 3.6.

Observation Number	*M1* Input	*M2* Input	Output	Hypothesis 1 Fault in $M1$ (parameter value)	Hypothesis 2 Fault in $M2$ (parameter value)
1	1.013	1.987	10.082	3.987	4.013
2	2.016	1.006	8.003	2.471	3.944

Table 3.6: Results with noisy measurements.

If a comparison is made between this table and table 3.3, the results are seen to be very similar, the problem is that these new results are not exact. If the final column in the table is examined, the two values shown are both approximately equal to 4 and therefore the fault could be in $M2$ with the parameter changing from 3 to 4. But it is not possible to be as certain as before. This time we are using our own judgement

and feelings to say that the results are 'close enough', the crucial question is 'how close is close enough?'. If the noise levels on the signals were higher, then it would become more and more difficult to decide in which component the fault is or indeed whether the fault could be explained by a failure in just one of these components at all. The converse problem also exists; when noise is present it is always possible to say that the results are not consistent and they therefore do not explain the fault, as in the last column of table 3.6, '4.013 does not equal 3.944 so a fault in $M2$ does not explain the fault'. This is a major problem with constraint propagation/satisfaction in its basic form, it is built upon the principle of making signals consistent and noisy measurements and estimates cannot be made consistent! This problem is addressed fully in later chapters, but constraint propagation/satisfaction methods alone are inadequate for tackling the problems of noise.

The implications for the actual detection of a fault should be apparent. A fault is detected by measuring the inputs and outputs and then seeing if they are consistent with the model of the system. If the measurements contain noise then they will never be consistent. The ability to ignore small differences in the signals is required. To do this some sort of threshold has to be set, differences below the threshold are due to noise and differences above it are due to a fault. This threshold may use the result of a complex algorithm to decide if a fault is present, but somewhere one has to decide what difference in signals is allowable and what difference is suspicious. This will mean that detecting a small fault will be difficult and there will be a delay in detecting faults because it is not clear whether the difference in the signals is due to noise or a fault. Also, with noise comes the possibility of having false alarms. If the threshold is set too low, then the noise in the signals may be viewed as suspect and a fault may be indicated. If the threshold is set too high, then genuine faults may have their detection delayed or missed altogether.

In systems that contain dynamic components the situation where noise gets amplified as signals are propagated through the system can arise. If these signals are then used to estimate parameter values then the estimates may contain very large errors. Some method to reduce noise levels will be required. These noise reduction techniques have been used by many others but not in the area of constraint propagation and suspension. [19] [69] [31] [72] [26] [80] [79].

3.9 Dynamic Components

Dynamic components are components whose output is a function of its past inputs, for example a water tank. The level of water in the tank is a function of the past flows in and out of it. $Level = \frac{1}{cross\ sectional\ area} \int (\Sigma flows)\ dt$.[2]

These are usually the states of the system.

Unlike non-dynamic components, it is also necessary to consider the current state of the component as well as its input, output and parameter. Of the work that has been done using constraint propagation in dynamic systems, references [52] [51] assume that all states in the system are directly measurable, effectively transforming the problem into a static one. Doing this is possible in a number of situations, but there are many where all of the states will not be measurable. In these situations an estimate of the systems states must be made. It will be shown later that this task is complicated by the presence of noise on the signals.

The constitutive relationship of a typical dynamic component is $output = \frac{1}{P} \int input.dt$, where P is the component's parameter. The model used here is a discrete time model of the system and so $output = \frac{1}{P} \Sigma input \times dt$ will be used, where dt is the time interval between samples of data taken from the real system, and *input* is

[2] $(\Sigma flows)$ means the sum of the flows in to and out of the tank through each of the pipes connected to it, at any instant of time.

the value of the input to the component at each sample. Alternatively *output* = *last output* + $\frac{1}{P}$(*input.dt*) or *output* = $\frac{1}{P}$(*state*) where *state* = *last state* + *input.dt* could be used. This requires the last value of the output from the component (the state from the previous sample time). This causes problems: to find the present output from the component, the present input and its previous state are required, but the previous state is a function of the previous input and an earlier state. This situation is illustrated in figure 3.18, here, to find the output, the present input and previous output are needed. Without the value of the previous output nothing can be done. If the input and the output were measured, the last value of the output measured and the current value of the input could be used to estimate the present output from the component. This could then be compared with the current value of the output that has just been measured in order to detect the presence of a fault.

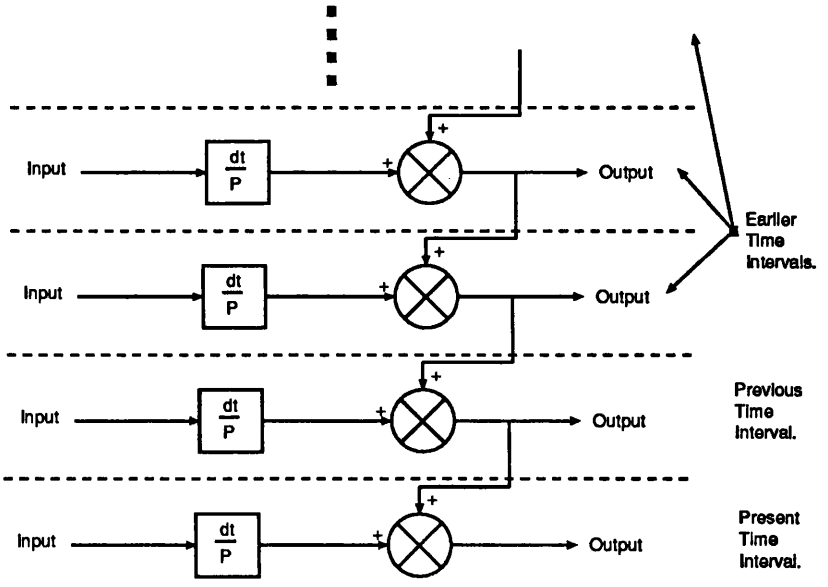


Figure 3.18: The present output depends on all the previous outputs.

It is possible to propagate signals in the opposite direction to that described above, that is use the current and previous measurements of the output and then propagate these through the component to find the input, and then compare this with the measured value for the input. Doing this in the presence of noise however results in

very poor estimates of the input. This is illustrated in figure 3.19.

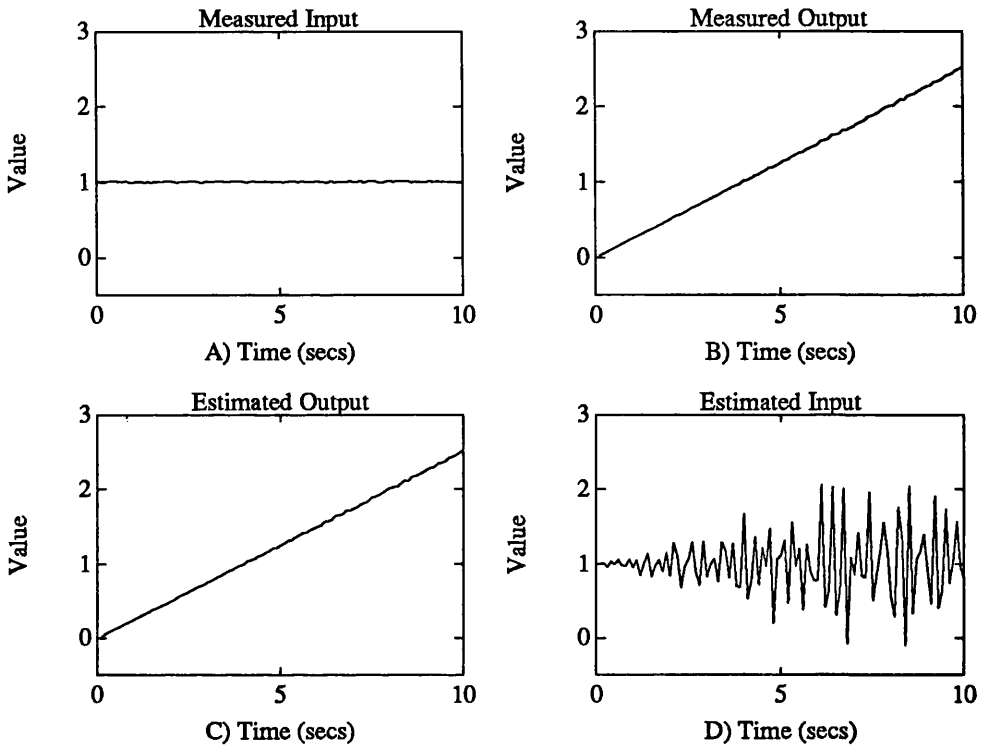


Figure 3.19: Propagating noise through dynamic components.

Figure 3.19(A), shows the measured input from the system of figure 3.18. Samples are taken every $\frac{1}{10}^{th}$ of a second. The value of the input is 1 with a 1% white noise to signal ratio. Figure 3.19(B) shows the measured value of the output, it starts at zero and gradually rises with time, because the output is a summing function of the inputs. The components parameter P is 4. Figure 3.19(C), shows the estimated value of the output, using the method described above i.e. using the current value of the input and the previous value of the output to calculate the current value of the output. It can be seen that it is very close to the measured output. Finally, figure 3.19(D) shows the estimated value for the input. This time the estimate is very bad, although the average estimate can be seen to be about 1, the actual estimates themselves are very noisy. This was due to the signal being differentiated. If a larger system was being dealt with, which had a second similar component. Then estimates could be

propagated through that as well. The results of this would be very close to nonsense.

In summary.

- Dynamic components have the added problem of being a function of previous system inputs. This means it is necessary to keep a track of how the system was at the previous time interval, the ability to estimate the states of the system is also required. This can become difficult when the system may have developed one or more faults.

- Propagating noisy signals through dynamic components can amplify noise and make results meaningless, however, to perform diagnosis on real system there is no choice but to use noisy measurements and to propagate signals backward through dynamic components. Some form of noise reducing methods will be required.

3.10 Combinatorial Problem of Multiple Fault Diagnosis.

In a system which contains many components, the number of possible combinations for components failing simultaneously (i.e. the number of possible hypotheses) is immense. Consider a system which has four components, named A, B, C and D. There are 15 different combinations of components which could fail simultaneously. These are listed below.

1. {A}.	4. {D}.	7. {A, D}.	10. {C, D}.	13. {A, C, D}.
2. {B}.	5. {A, B}.	8. {B, C}.	11. {A, B, C}.	14. {B, C, D}.
3. {C}.	6. {A, C}.	9. {B, D}.	12. {A, B, D}.	15. {A, B, C, D}.

For a system with 5 components there are 31 different combinations of possible faults.

If there are N components in a system then the number of combinations of 1 or more components failing simultaneously is $2^N - 1$.

There is a relatively large amount of computation required to check whether one individual hypothesis accurately describes the behaviour of a faulty system or whether it does not. In a large system with many components it is clear that testing every possible hypothesis is not practicable in real time. This is one of a number of reasons which makes diagnosing multiple faults much more difficult than single component faults. Another aspect which makes multiple fault diagnosis difficult is the problem of how to test a hypothesis which assumes that a large number of components have become faulty. If it is assumed that a number of components are faulty then the amount known about the system is reduced, and it can quickly become difficult to reason about a system which is only partially known. This problem will be discussed in detail in chapter 6.

It is clear that testing every hypothesis in real time with a model based diagnosis method is not practicable; either some method other than model based must be found, or a way of adapting the model based diagnosis method, in such a way that there is no need to test every hypothesis, must be found. Because of the advantages of using a model based diagnosis method (see section 3.1) a method which reduces the number of hypothesis which are tested to N, but which also allows us to reason about the hypotheses which were not tested is presented. The overhead of this additional reasoning is very small when compared to the amount of computation required for testing a hypothesis and so it becomes possible to diagnosis multiple faults in real time using this method. In this method, sets and subsets of multiple faults are teated in a similar way to that of de Kleer & Williams [10] [11] and Reiter [68]. The algorithms

which handle these fault sets are, however, original as they are required to deal with imprecise system models, noisy systems and recursive in time diagnosis.

This method of reasoning is based on some of the work of Reiter [68], but the same method of diagnosis is not used, as it is not suitable for this kind of real time application due to the the computational overheads which would be introduced using a recursive in time diagnosis method. However, his work is built upon and a similar approach of examining subsets of hypotheses to reason about other hypotheses is used.

Reiters method is based upon finding a subset of all of the system components which, when all of these components are considered to be behaving correctly, are consistent with observations taken from the system. All the system components which are not in the above subset can be regarded as being a diagnosis of the fault which has appeared in the system, i.e. all of these components can be regarded as being faulty.

A fault diagnosis is a set of components which when regarded as being faulty, will explain the observations. However, any superset of these components will also explain the observations. So the problem is to find all of the sets of components which when considered to be faulty will explain the observations, and which have no proper subsets which will also explain the observations. It is possible to arrive at more than one diagnosis which meets the above criteria.

To check a hypothesis one needs a fault hypothesis checking routine. This will vary drastically according to which domain the diagnosis is taking place in; routines which check a fault hypothesis for a digital electronic circuit and for a chemical manufacturing process will be very different. A number of observations of the system need to be taken, together with a set of components which are assumed to be working correctly. These are then passed to the routine which must then check that the

observations and the set of working components are not inconsistent[3]. If they are not inconsistent then the other components (the ones regarded as being faulty) are a diagnosis, but not necessarily a minimal one, i.e. without subsets which are also diagnoses.

Reiter presented a method of deciding what was the best subset of components to check for next. Using a tree structure of nodes consisting of hypotheses, the method, depends upon its ability to check a hypothesis with a set of observations. This conflicts with the objectives of having a real time fault diagnosis algorithm, because to check a hypothesis many observations may be needed, as component parameters need to be estimated and identified. This would mean waiting until there are presumed to be enough observations and then starting to check hypothesis. If there were too few observations, poor results would be obtained and it will be necessary to wait longer. If there were enough, then maybe good results could have been obtained earlier. A recursive in time fault diagnosis algorithm is required. This algorithm can utilise Reiters ideas about manipulating sets of hypotheses, but cannot implement them in the same manner as he proposed because of the different requirements. In chapter 6, the problems of dealing with multiple faults are looked at in greater detail.

[3]This means that the routine is unable to prove that the observations are inconsistent, rather than it being able to prove that the observations are consistent.

Chapter 4

Constraint Propagation in Dynamic Systems.

In this chapter the method used to propagate signals around a dynamic system and through time to enable the determination of values for signals, parameters and states inside the model will be looked at. This is in principle the same as that done by Leary [51], and Leary and Gawthrop [52], but this is extended to cover unmeasured states, multiple component failures, propagating signals through time, propagating constraints simultaneously and automatic system constraint generation. The problems caused by propagating noisy system measurements through a model which contains dynamic components will be examined, and details will be given of the pre-filtering introduced to reduce the effects of the noise. A method which will be used for estimating the systems states and estimating the parameters of components will be described. Finally the problems of using these techniques in real time with noisy measurements, and how these can be overcome, are discussed.

In chapter 3, it was demonstrated how to propagate measured signals through a model of the system, and thereby find the values of signals inside the system. This information is then used to find a value for a component's parameter which

make these measurements consistent. In this chapter dynamic systems are looked at in more detail and an automatic propagation algorithm used to analyse systems is introduced.

In figure 3.18 a brief view is given of how measurements from a number of consecutive observations are used to enable the evaluation of signals in the model of the system, when the system contains dynamic components. Now, considered in more detail, is an example of a dynamic system. A dc motor is examined and it is shown how it is possible to:-

1. find a component's parameter,
2. find the parameter of more than one component simultaneously,
3. find the states of the system, and their rates of change,
4. use the model to find the expected values for the outputs of the system,
5. and how the number and location of sensors on the system affect the ability to find signals, states and parameters in the systems' model.

The examples given here will be illustrated with reference to block diagram models of systems. It should be noted that this is for illustrative purposes only, all the software operates on bond graph models of systems. Block diagrams are a clear way to explain the features and principles involved, but bond graphs are easier to manipulate.

We begin by looking at constraint propagation, some of its common problems and some solutions to these. It is then shown how these constraint propagation techniques can be used to find the states of a dynamic system, the value of a component's parameter and the rates of change of the system's states. In addition it is shown how constraint propagation, together with the causality assigned to the bond graph model

of system, are used to enable the prediction of what the outputs of the system should be.

Attempting to implement an algorithm to perform constraint propagation in real time is only possible if the system is small and simple, and even then considerable computing power would be needed to use the algorithm for real time fault detection and diagnosis. One method for overcoming this will be examined. This method results in a reduction of the computational load as a whole. The majority of what is left is shifted away from being done in real time, and is instead done before monitoring a system for faults begins. This requires analysis of the system by constraint propagation algorithms and then solving simultaneous equations to reduce the real time processing needed. A way of doing this automatically for a wide range of dynamic systems will be shown. Finally one technique for pre-filtering the measured signals is looked at, this is the first step in reducing the effects of measurement noise in the detection and diagnosis algorithms.

4.1 Propagating Signals through Time and the Model.

One of the problems faced when looking at dynamic systems is the problem of finding what state the system is currently in, since the current state depends on the past history of the system inputs and its states. To solve this problem signals from past time intervals will be propagated into the current period of time.[27] This method appears similar to the "back propagation in time" method used in neural networks when training them on dynamic systems. Both methods are essentially attempting to do the same thing, which is to resolve the dynamics in the system by using information from previous time periods.

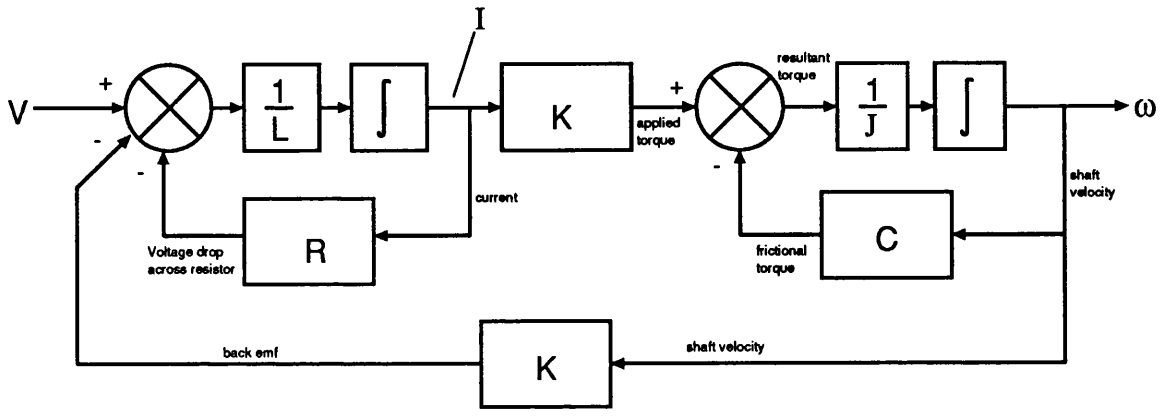


Figure 4.1: A block diagram of a dc motor.

Figure 4.1 shows a block diagram of a dc motor where:-

- **R** is the motors electrical resistance.
- **L** is the motors electrical inductance.
- **J** is the motors inertia.
- **C** is the friction generated in the motors bearing.
- **K** is the constant between the current flowing through the motor and the mechanical torque generated, and the velocity of the shaft and the back emf.
- **V** is the voltage applied to the motor.
- **I** is the current going through the motor.
- ω is the angular velocity of the motor's shaft.

As discussed in section 3.9, it is not possible to propagate through dynamic components in either direction unless the value for the state of the dynamic components at the start of each observation (or time period) is known. From just

one value of V and ω it is not possible to propagate throughout the whole model of the system. Figure 4.2 show the result when an attempt is made to propagate from only V and ω. The signals (arrows) which are shaded indicates where propagation was possible.

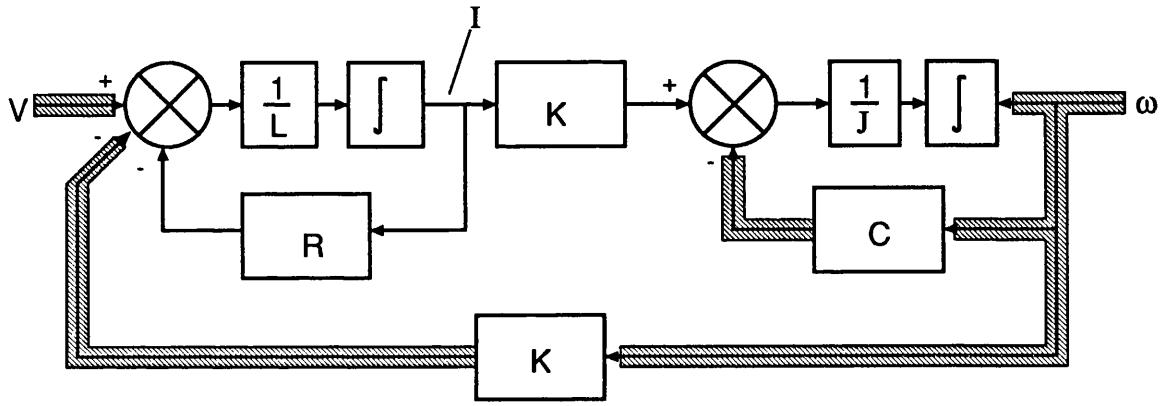


Figure 4.2: The extent of propagation from V and ω.

If propagation and suspension techniques are to be used for dynamic systems then more information about the previous values of the states must be used. Signals must be propagated from previous observations of the system to provide the initial values of the states. These are needed for propagating using the current observation. This is done as follows :-

In figure 4.3 three block diagram models of a dc motor are shown. V_1 is the voltage being applied during the current period of time. V_2 is the voltage at T = -1, the previous time period, and V_3 is the voltage at T = -2. Similarly for ω. The relationships for the states are in their discrete time form of $state = laststate + \frac{1}{p}(rate.dt)$. Here, these models represent the dc motor during three consecutive periods of time. The models are connected to each other via the states, the current value of each state from one model is connected to the previous value of

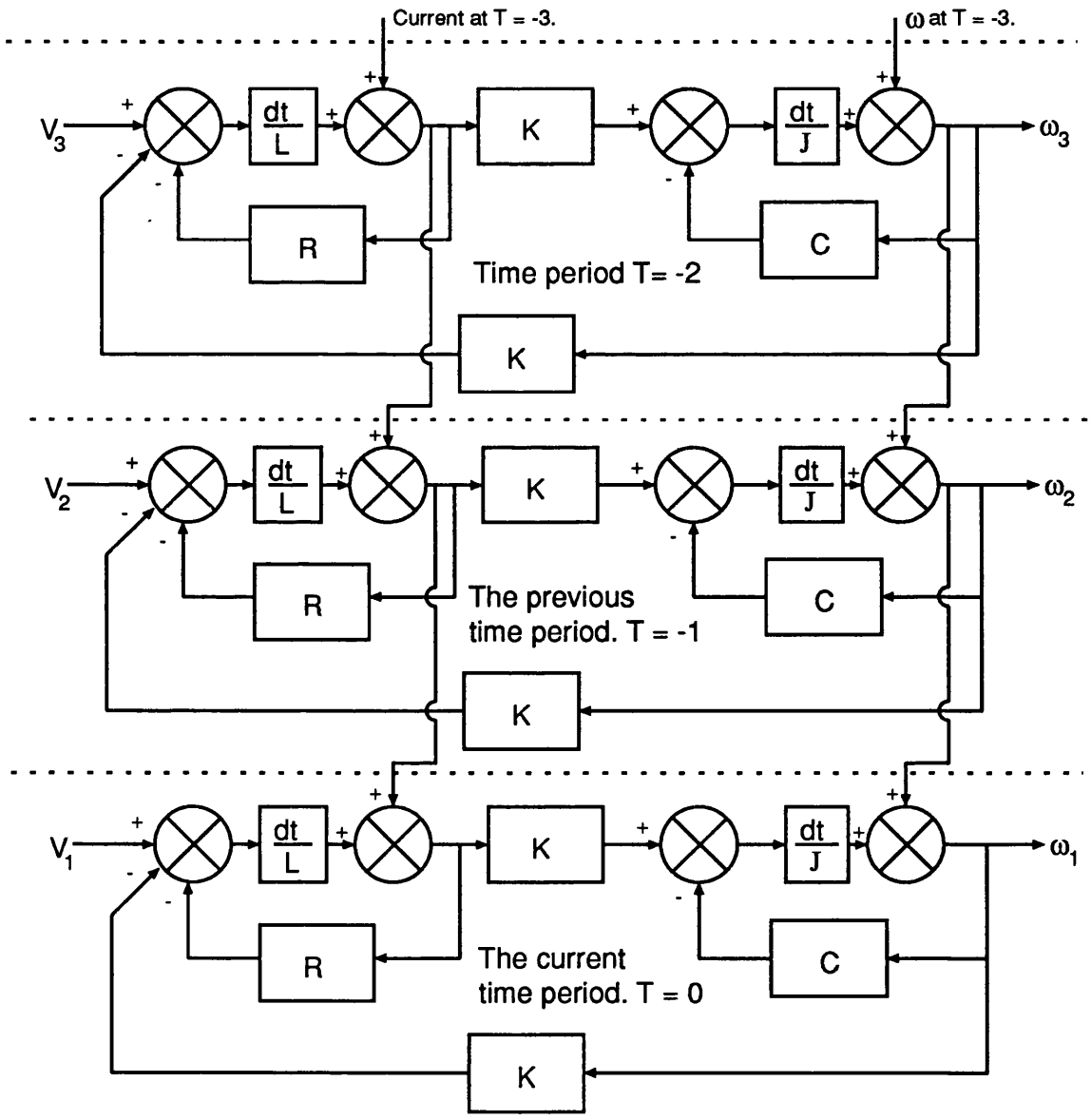


Figure 4.3: A dc motor model for three consecutive time periods.

the state for the model at the next time interval. If one takes the model of the system as a series of models connected together to represent different periods of time, then it is possible to propagate signals through time as well as the model and the dynamic behaviour of the system can be captured.

If the case was that V, ω and I were measured, then it would become possible to propagate signals throughout the system without using multiple time step models. However, if any component was suspended so that its parameter could be estimated, then it would not be possible to propagate completely and a value for the parameter would not be found. One would then be forced again to use multiple time step models. In general, the more sensors that are measuring a systems behaviours, the fewer the number of consecutive time step models which have to be used. This also depends upon the location of sensors and the topology of the system.

4.2 Constraint Propagation.

In figure 4.2 it was shown that it was not possible to propagate signals from V and ω throughout the model of the system, using just a single observation. Figure 4.4 shows the same situation, except this time two observations of the system are used. The arrows show the way signals were propagated from V_1,V_2,ω_1 and ω_2. The basic path is as follows. Firstly ω_1 and ω_2 are used to generate signal *e* and hence *f*. *c* and *f* produce *g*, *h* and then *i*. Now *j*, *a* [1] and *i* are used to produce *k* and then *l*. At this point all of the signals for the current observation are known, and if that was all that was required, propagation could stop at this point. Alternatively propagation could proceed to find all of the signals in in the model from the previous observation. To do this *l* and *h* are used to get *m*, then *n* is found and *x* is used to find *o*, *p* and then *q* with the use of *d*. Also, from *m* *r* is found and then using *t*, *s* and *r* the values

[1] The signal from input V_1

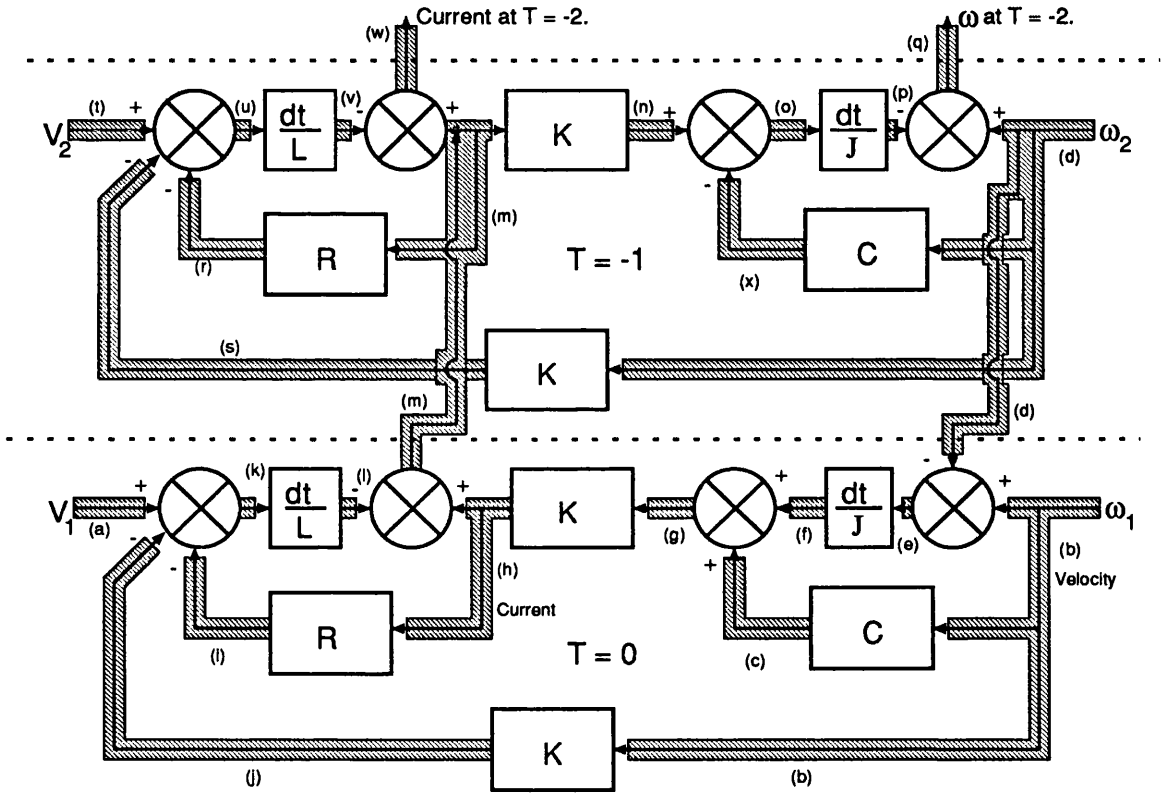


Figure 4.4: Propagating signals from V and ω throughout the model.

of u, v and finally w are obtained. Now all of the signals in the model of this and the previous observation are known.

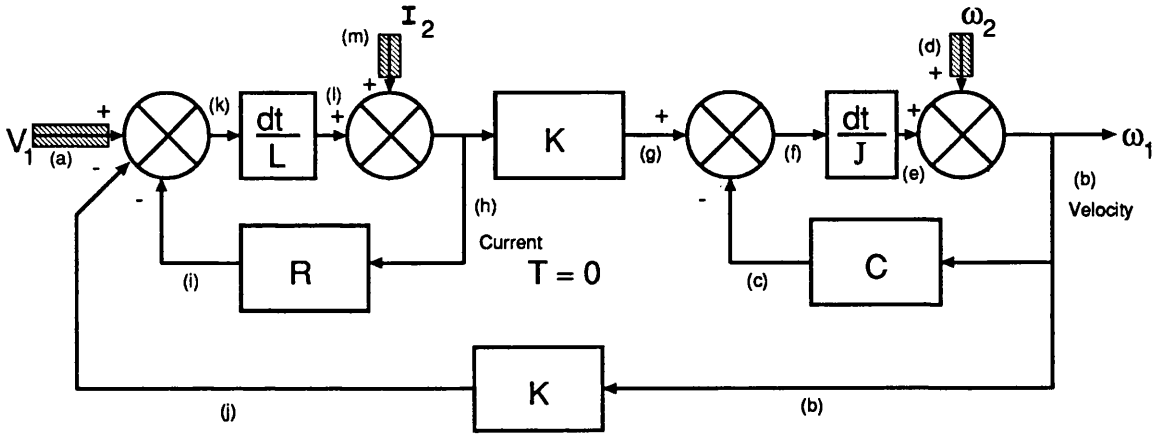


Figure 4.5: Propagating from the input and the states to the output.

The propagation of signals described above was straight forward. A signal was simply passed through a component to find another signal, and this was repeated until all the signals were known. Only one constraint (component) was considered at a time and the situation where there were no signals that could be used for propagation never arose. Now consider figure 4.5, here the input V is known, as are the values of the states at the start of this time period I_2 and ω_2. It is desired to find the output ω_1. This should be straight forward but it is found that it is not possible to propagate any of the signals. Since j and i are not known, neither k nor l can be found. Since l is unknown h can't be found and since e is unknown b cannot be found. What must be done here is to consider more than one constraint simultaneously. Conventional propagation algorithms are unable to cope with simultaneous propagation, but now CLP($\Re$) [42] and other programming languages [53] can deal with this, at least when the resulting equations are linear. Using these techniques and a mathematical solving package, it is possible to deal with a limited range of non-linear simultaneous

equations.

The equations of the individual components are shown in table 4.1 and are the constraints of the system. There are three known signals, a, m and d and there are ten unknown signals, b, c, e, f, g, h, i, j, k and l.

1.	$a - j - i = k$	6.	$g - c = f$
2.	$k \times \frac{dt}{L} = l$	7.	$f \times \frac{dt}{J} = e$
3.	$l + m = h$	8.	$e + d = b$
4.	$h \times R = i$	9.	$b \times C = c$
5.	$h \times K = g$	10.	$b \times K = j$

Table 4.1: The constraints of the system.

As can be seen there are ten equations which describe the system, and ten unknown signals. These equations can therefore be solved simultaneously and b can be found, which is equal to ω_1. Although this does not seem the same as propagating signals through the components, it is essentially the same, the only difference is that more than one constraint is considered at a time. In this case all of the system constraints needed to be considered because of the feed back loops, but in another case it may be possible to propagate signals individually through components, then consider a number of constraints simultaneously and then propagate further signals individually. In all cases the results are the same. If there are more unknowns than equations then it is not possible propagate completely and one may wish to reconsider the number and location of sensors. If, however, there are more equations than unknowns, then propagating completely will be possible, but some information will be leftover. This extra information can be used to help to identify a components' parameter[2].

Using these techniques any signal in the system can be identified, provided that the equations can be solved where necessary. If the system has more states, then more observations of the system are needed [3]. An algorithm will be introduced later

[2]Identifying a components' parameter means its' current value is regarded as unknown, i.e. another unknown has been introduced. To solve this more information is required .

[3]Alternatively it could be said that more consecutive in time models of the system connected

which will work out how many observations are necessary to evaluate the values of any combination of unknown signals or parameters.

There are a number of functions which will be required by the fault detection and the fault diagnosis algorithms. These functions yield internal signals from the system's model, based on known values for some of the other signals. For example, the system outputs as a function of the system inputs and the initial value of the states. The reasons why each of these functions is required will be given in chapters 5 and 6, but the way these functions generate their results will be given here.

4.2.1 Finding the States of the System.

Finding the system states is a relatively straight forward process. A route is simply found along which signals can be propagated from the states to the signals which are measured. So, looking back to figure 4.4, the states are the current h and the velocity b. Firstly b is known because the speed of the motor is being measured. To find h the values of ω_1 and ω_2 are propagated forward along d, b, e, f, c and g to give h, the current.

In the fault diagnosis algorithm described in chapter 6, it will also be required to find the states at the start of the current observation, these would be m and d in figure 4.4. These can be found in a similar way as described above except propagation would take place from ω_1, ω_2 and V_1. This could be described by saying 'what were the values of the states at the start of the current period as a function of the current observation and as many previous observations as necessary'. In this case only the current observation and 1 previous observation is needed. In a system where there are more unmeasured states the information from further previous observations would be needed.

---

together are needed.

Also required by the diagnosis algorithm will be the current values of the states as a function of the rates of change of the states and the current observation[4].

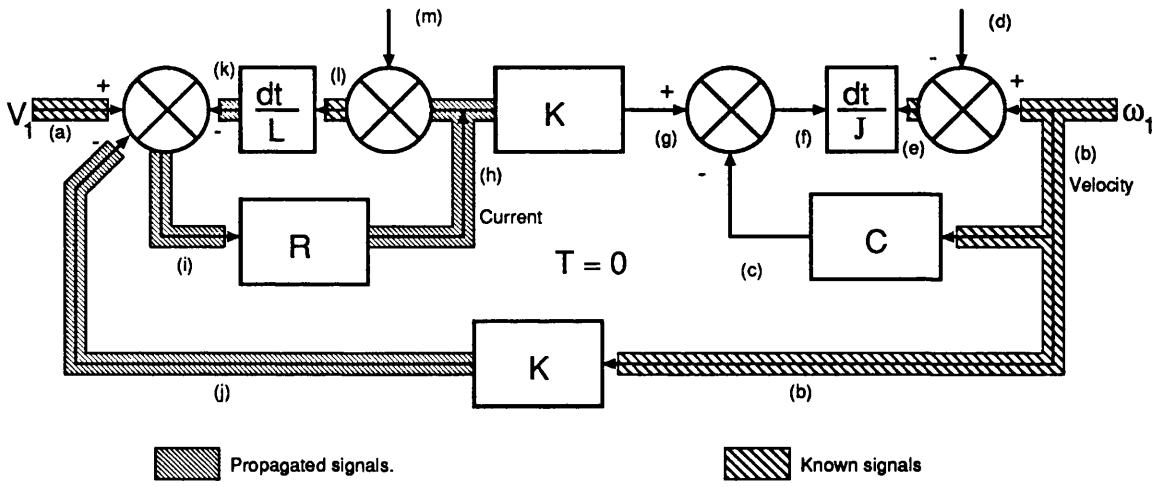


Figure 4.6: States as a function of the current observation and the rate of change of the states.

This situation is shown in figure 4.6. Here, V_1 (a), ω_1 (b), l and e are known, and the two states h and b are to be found. It should be noted that l and e are not the actual rates of change of the states but are in fact the change in the output of the state component, these signals are known because, as described later, they are estimated and filtered and then used in finding the value of the states. Finding b is easy as it is equal to ω_1 which is measured. To find h propagation take place from l to k, and from b to j, and then a, k and j are used to gives i, which is then used to find h, the state. The current states as a function of the change in the states and the current observation has now been found. These propagation routes are worked out automatically by an algorithm which will be described later in this chapter.

[4]Previous observations may also be used as well if it is necessary

4.2.2 Parameter Evaluation and Constraint Suspension.

There are two functions which will be used to estimate parameters. Firstly, a function which will yield the parameter values of one or more components which make a number of observations consistent. That is a function that will take a number of observations of a system and will then find the parameter values of one or more specified components which will make these observations consistent. As an example, suppose the value of the resistor R is unknown in the dc motor. Three observations of the system would be needed and a model of the motor which consisted of a dc motor in three consecutive periods of time would be used. These observed measurements would then be propagated throughout the model and a value for R which would make all of these observations consistent would be found. Three observations are needed because there are two states in the system and an unknown parameter. This is shown in figure 4.7.

This is based on the principle that when component R is suspended and then signals propagated throughout the model until a signal arrives on either side of the component, then these signals can be used to find a value for R's parameter which makes these signals, and hence the observations, consistent.[52] [51] If the values for both L and R were sought, four observations and a four time periods model of the system would be required. To find the value for C's parameter would essentially mean doing the same thing, but a number of constraints would have to be solved simultaneously. This does not pose a problem as long as these equations can be solved, this will be described in detail in section 4.4.

The second function needed is similar to the above except only one component at a time is suspended, and only the current observation and the current changes in the states are used (as described earlier in section 4.2.1). What is of interest are

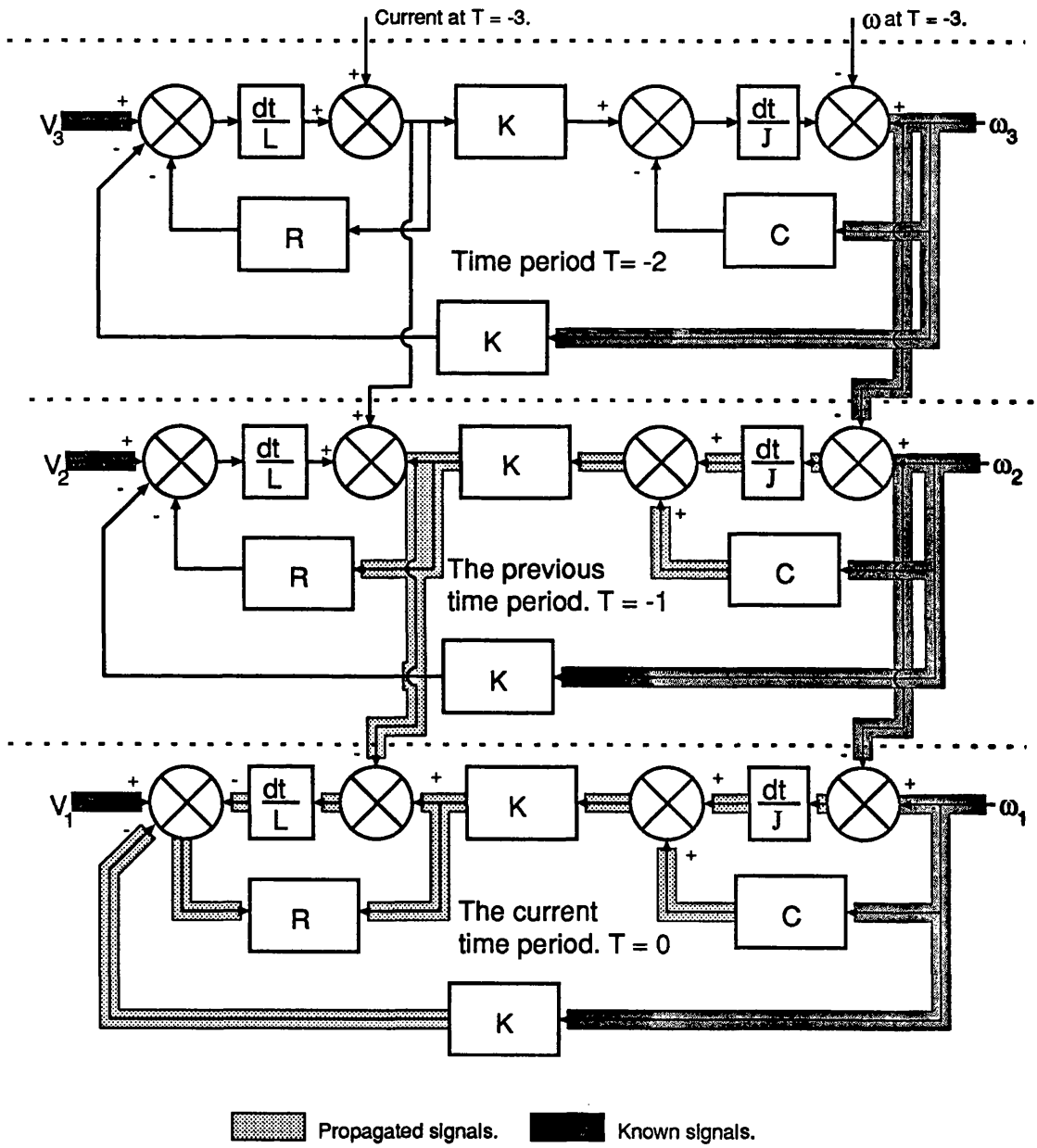


Figure 4.7: The system model for three observations.

the two signals which arrive on either side of the suspended component. The use of these will be fully described in chapter 6, but in summary these signals will be used to estimate the component's parameter. If the component is linear then these signals are simply used in a least squares type algorithm to estimate the parameter. If the component is non-linear then two values whose ratio equals the components parameter are required. For example, if the input to a component is X and its output is Y, and its constitutive relation is given by $Y = P \times \sqrt{X}$ where P is a parameter, then Y and $\sqrt{X}$ would be used in the least square estimation, rather than Y and X.

4.2.3 Finding the Rates of Change of the States.

Another function that will be required is a function which will give the change in a state from the previous to the current observation. There is a condition here that the change in a state must be generated by first finding the current and previous values of the state. Referring to figure 4.4, it can be seen that there are two ways in which the change in current (l) could be calculated. Firstly propagating from ω_1 and ω_2 to g, h and then i, propagating from ω_1 to j, and then using a, j and i to propagate to l. This method did not involve finding m, the previous value of the state, and so does not meet with the condition stated above. The second method for finding the change in current (l) is to use ω_1 and ω_2 to find h and use ω_2 and ω_3 to find m and then taking the difference between h and m.

It was found that if the first method for calculating a change in the state was used, large errors were obtained due to the effects of summing signals of very different magnitudes. Although calculating changes in the state by finding the difference between the current state and the previous state also produced errors, it was found that the nature of the errors over time was equivalent to noise, whereas the error produced by the first method was found to be an offset with the addition of noise.

The offset was due to the summing of different size signals. For example, looking again at figure 4.4, there are two summing components (not counting the summing effect of the state relations). In using the first method, the first summing component arrived at is associated with the torque on the rotor, f is the rate of change of the momentum derived from ω_1 and ω_2, this is added to c, the frictional torque, giving g, the total torque coming from the motor windings. If the speed is high and changing slowly, then c will be large and f will be small, resulting in g being approximately equal to c. Part of this is caused by the limited floating point resolution of computers, but even if the floating point resolution is high, f can still be made insignificant by even low levels of noise on c. When g is then used via h and i in the next summing component, the result, k, will be missing at least some of the effects of the slow change in speed of the motor. The rate of change of the inductor will always be slightly biased in one direction when the motor is speeding up, or biased in the other when it is slowing down.

4.2.4 Finding the Outputs of the System.

This function has to be capable of calculating what an output from the system should be using measured data from one observation, and the initial value of the states at the start of the observation[5]. There is one of these functions for each and every output from the system. For each system output the following is carried out. Firstly a model of the system is used for just one observation. Estimates of the initial value of all the states at the start of the observation are used together with all of the inputs and outputs from the system, that have been measured for the current observation, except the output which is being calculated. Propagation then takes place forward from these known values, following causality, until the output in question has been reached. This

[5]An output is any measurement from the system where the value of the measurement is affected by the behaviour of one or more of the system's components.

produces an estimate of what the output should be based on the measured signals from the system, the state estimates, and the parameters of the components used in this calculation. This is repeated for each output of the system.

As will be shown in more detail later, these output estimates can be compared to the measured outputs to check for the presence of a fault or to check the accuracy of a hypothesis during diagnosis.

An alternative way of doing this would be to use the states and all of the input measurements and then, from these, calculate all of the outputs at once. This was found to be unsatisfactory for a number of reasons.

1. Calculating all of the outputs from the states and inputs can be complex and require a significant amount of processing time, just for performing the arithmetic. Using all of the available measurements except one output to calculate a predicted value for the output results in smaller calculations while still ensuring that any fault will appear in at least one of the system's outputs.

2. The propagation from the input to the output can be complex, requiring a number of simultaneous equations to be solved, if these equations happen to be non-linear, then often unique solutions are not possible. Propagating backwards from one unknown output until any of the measurements taken from the system are reached is quicker, often not involving simultaneous equation solving and keeping non-linear components in relatively smaller systems of equations.

3. If there is a slight modelling error, or a fault just after one of the inputs, then its effects can be amplified when propagating signals from it, throughout the rest of the model. This results in the system being sensitive to faults and modelling errors near the inputs but less sensitive further away. Although the system should correctly detect faults, it should not be over sensitive and therefore create false alarms. The system should also be robust with respect to modelling errors.

That is, an even and controllable sensitivity would be more desirable than being very sensitive to some of the system components and less sensitive to others. Using the method outlined above, the need for long propagations from the furthest input to the furthest output is reduced and this keeps the amplifying effects to a minimum, making the sensitivity to particular components more even.

4.3 Real Time Propagation.

The functions described above involved using known signals and passing these through components to find other signals and then repeating this until the required values have been found. The actual method for doing this requires keeping a list of the known signals and components which have not yet been used for propagating signals, these components must be searched to find which ones can be used next and which signals would be produced by doing so, the new value of a signal needs to be worked out, component must be removed from the list and then all of this must be repeated with the new component list.[38] [36] [8] [51] [52]. Complications can also arise when trying to satisfy constraints simultaneously, either by too many unknowns or by constraints which are too complex and are unable to be solved. Obviously, trying to do all of this in real time while monitoring a system will often not be practicable with the current computing technology[6].

If, for example, a system was being monitored, and measurements were taken say 20 times per second, and these measurements were then propagated around various different models, filtering data and estimating parameters, then it is clear that the processing capability necessary is enormous and indeed much of it, like deciding which way to propagate, would be needlessly repeated for every observation. To use these

[6]That is not to say that it is impossible, merely prohibitively expensive.

methods in real time, it is necessary to shift as much processing as possible away from real time, while still ensuring the system can detect and diagnose faults in real time, as they occur. This is done here by doing some off-line pre-processing. That is, analyse the system, work out the best routes for propagating signals, solve simultaneous constraints and identify which faults are diagnosable and which are not. To diagnose a fault[7] the faulty component parameters need to be estimated, if there are too many unknowns to solve these equations, or the equations are too complex to solve uniquely, then diagnosing faults occurring in these components will not be possible. This will be discussed later, but it is important to note that there is no point spending 'real time' trying to solve equations which cannot be solved.

As an example to clarify this and to show the advantages, consider again figure 4.6. Here h is found by propagating a, l and b along the shaded arrows. To do this a model of the system is needed which contains the relationships of each component and the structure of the system, i.e. how the components are connected. Also required is an algorithm to perform the propagation, work out the values of signals, decide which signal to use next, until the objective is reached. Alternatively, this could all be done before hand, equations could be solved and one line of code in a program could be used which simply gives the value of h as a function of the parameters and the known values, in this case

$$h = \frac{a - l\frac{L}{dt} - K.b}{R}$$

and the line in the program would be

```
h = (a - l * ( L / dt) - K * b) / R ;
```

This one line of code can replace the whole model and the propagation algorithm in the on-line detection and diagnosis software. Whenever this state is required as a

[7] A fault may consist of more than one component failing simultaneously.

function of the measurements and the change in states, this line of code is used. To make it easier to combine with the rest of the software, the line of code is encapsulated in a function, and when it is needed the function is simply called. This is done with all of the propagation schemes described in the previous section, the result can be a large number of functions if the system is large, but the result is also a massive increase in execution speed of the on-line software.

In order to do this off-line analysis which can create many functions, it is necessary to have some software which will do all of the analysis automatically, as doing it by hand will be time consuming, tedious and prone to error. In the next section an overview of the off-line software which has been developed will be given, together with some details of the propagation algorithm and how it functions.

4.4 Automatic Constraint Generation.

An algorithm has been developed which will automatically generate specified functions from a system model as described in the previous section. This has been implemented in conjunction with Model Transformation Tools (**MTT**), which is a bond graph manipulation and analysis toolbox[28] [24] [22] [34] [32]. A graphical picture of a bond graph is taken as input, together with a short definition file, and after a number of transformations a complete C++ code program is automatically produced which can be compiled and which is able to detect and diagnose faults in a particular dynamic system described by the original bond graph. An outline of this process is given below.

1. The algorithm begins by taking a bond graph model of the system, it is informed which signals to find and which signals are known. For example, 'I wish to find OutputA. InputA, InputB, OutputB and StateA are known.' This is equivalent

of saying 'Find OutputA as a function of InputA, InputB, OutputB and StateA.' The unknown does not have to be a signal, it may be a component's parameter.

2. The algorithm will now find the shortest route from OutputA to the signals which are known. If the route involves going through a state component, then its previous state must also be found and so the model of the system from the previous time step is used as well.

   If more than one parameter is being found at once, it is necessary to make sure that propagation does not inadvertently take place through a component whose parameter is not known. Doing this means that one too few equations will result at the end of the propagation to solve all of the constraints since an additional unknown will have been introduced. If this must be done, then a further time period must be used and the additional information used to find another propagation route to the component with the unknown parameter. This will yield a second way to find this parameter. These two ways of calculating the parameter can now be solved simultaneously. If two or more parameters are being found, then this is repeated, descending to an earlier time period each time these circumstances occur. (This is explained in greater depth in appendix A.)

3. Having found a route from the known signals to the desired signals, the next step is to go along this route and list all of the constraints which signals must pass through. Considering the example in section 4.3, these constraints would be as show in table 4.2.

1.	$j = b \times K$	3.	$i = a - k - j$
2.	$k = \frac{L}{dt} \times l$	4.	$h = \frac{1}{R} \times i$

Table 4.2: The individual constraints passed through.

4. These constraints are then sent, in symbolic form, to a symbolic equation solving package. The package used here is called *REDUCE*. This package can handle a large number of equations and manipulate them symbolically, solving simultaneous equations to find a variable in terms of any other variables. In the example above the equations did not have to be solved simultaneously, but if they did, *REDUCE* would have done this for us automatically. The output of *REDUCE* is a text file which contains the symbolic relationship between the unknown signal and the known signals in the form *unknown signal* = $f(known\ signals)$.

5. This symbolic solution can be easily converted into a line of C++ code, or any other programming language, and then encapsulated in a function for inclusion into a larger program.

The above steps, 1 - 5, are used for every function which is required by the fault detection and diagnosis algorithms.

When attempting to find the parameter of more than one component at once, it may be that there are too many unknowns and it is not possible to find enough equations to solve these due to the nature of the system and the location and number of sensors. In this case the equations cannot be solved, and faults in those components cannot be diagnosed. Alternatively, there may be enough equations, but the solution is either not unique, or too complex for *REDUCE* to solve, particularly if the constraints need to be solved simultaneously. In this case again one cannot diagnose faults in these components without resorting to other methods. This situation usually only affects the diagnosis of multiple simultaneous faults, and it can be resolved by having more sensors on the system being monitored.

4.4.1 Finding the 'Best' Route for Propagation.

Here the method of operation of the algorithm is described. It is used for choosing the most appropriate route for propagating from known signals to any unknown signals or parameters which are of interest.

The propagation route is the path from signals which are known, to the signals which must be evaluated. That is, following this route from the known signals, and using every constraint that is passed through, the value of the signals which are of interest can be calculated. By the 'best' route, what is meant is the route which passes through the fewest number of components and which meets a number of specified criteria.

This will be illustrated with an example.

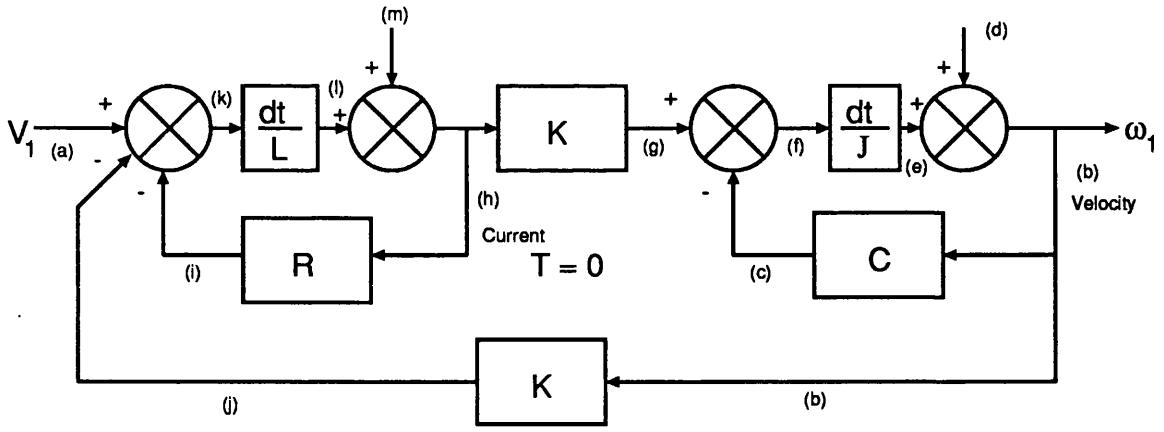


Figure 4.8: Automatic Propagation Path Generation Example.

1. The algorithm starts by locating one of the signals or parameters of interest. Suppose in figure 4.8 the value of h is to be found, given that all the parameters are known, and a, b, m and d are also known.

2. Next, one of the components that this signal is connected to is found. Lets take

this to be component **R** from figure 4.8.

3. Then all of the other signals which are connected to this component are found. The idea is that if all of these signals were known, then it would be possible to pass them through the constraint and find the signal of interest. In this example, 'all the other signals' consists of just i, as that is the only other signal connected to **R**.

4. After these signals have been found, a check is made to see which of these are already known, and which are not. The ones that are unknown are added to the list of signals which need to be found, and the signal of interest at the beginning is marked as being known. This is from the point of view that if the value of the other signals were known, then it would also essentially be known. It is only possible to go through a component once, but this can be done in either direction[8]. In this example there is only one other signal i and its value is unknown, the objective is now to find a value for i, since when this is done h will be found by propagating through **R**.

5. If, at this stage, all of these other signals are known, then a complete route has been found and this route will be used as the 'best' route.

At this stage a route has been found which consists of one step which will give the signal of interest provided that a second set of signals are known. Before the values for this second set of signals are found, it is first necessary to find if there are any other ways in which the signal could have been found, i.e. is the signal connected to any other components which could be used to derive its value. In the example, the other components which are also connected to h are **K** and the $\otimes$ summing part

[8]This is not always the case, sometimes it is desired to only propagate through components in a backwards direction with respect to causality. This will give information about which components caused the signal of interest to take the value it currently has.

of the inductors state relationship. If g was known, propagation could take place through **K** to find h, or if l and m were known $\otimes$ could be used and hence h. These are two alternative ways of finding h.

6. If the signal from step 1 is connected to another component then repeat steps 2 - 5 for this one, and then repeat this step for any other components that it is connected to. eg **K** and $\otimes$.

After this step is complete all the possible ways of finding the signal using just one step will have been found. The next stage is to find all the possible ways of finding the values of the signals from stage 3 above, for each of the single step routes described in step 5. But before this is done, the presence of repetitions in the routes must be checked. For example, if two routes go around a loop, one of which stated solving constraints A, B and then C, and the other stated solving B, C and then A, then obviously since A, B and C must all be satisfied, then the order in which they are satisfied is unimportant, and usually they must all be satisfied simultaneously so order is irrelevant. In this case one of these possible paths can be safely removed from the set of possible paths which are being investigated.

7. All of the above steps must now be repeated for each of the possible paths that are being investigated. In the example the above must be repeated for i, which is the unknown signal needed to propagate through **R**, and the above must be repeated for g which is the signal needed to propagate through **K** and the above must be repeated with l, which is the unknown signal needed to propagate through $\otimes$, (remember m was mentioned at the beginning as a signal which is already known).

Just before step 7 begins, all the possible ways of propagating through one component to find the signals of interest will have been found. For each of these there

is a set of signals associated with it which must be known in order to to calculate the desired signal. If all of the associated signals are known for any of these 1 component propagation paths then the algorithm is stopped here and this path is used as the 'best' way of finding the signal. If, on the other hand, not all of the associated signals are known for any of the paths, the algorithm proceeds to find all the possible ways of finding the desired signal by propagating through two components, and then three and four and so on until the first complete path from known signals to the signals of interest is found.

The possibility for the number of paths under consideration in the above algorithm to explode is great unless some precautions are taken, and even then it is still quite possible for the number of paths under consideration to be very large, although from a practical point of view, this is unlikely to become overwhelming as will be explained.

The first condition is that all paths which descend a level of time should be ignored unless there is no alternative route. This prevents the algorithm from traveling down a path which just runs backwards in time adinfinitum. Also, it means propagation stays in the current period of time and uses as many of the current measurements as possible.

The second condition is to always remove similar paths, i.e. those paths which contain the same constraints but are in a different order.

An optional condition, that is used for finding the outputs of the system by following the causal chain of components as described in section 4.2.4, is that only paths which travel backwards through causality can be followed. This will find which signals caused the values of the system outputs.

The only way in which the number of possibilities could become overwhelming is if a large system is being considered, which has many components which have more than two connections (ie number of inputs + number of outputs > 2), the system has

a large number of states and there are few sensors on the system. If this was the case then it would probably be fair to say the system has too few sensors to be able to do very much at all, and attaching more sensors would not be unreasonable.

In summary, this algorithm can be used with

1. a bond graph model of the system,
2. a specification of signals or parameters which are unknown and are of interest,
3. a number of signals which are known,
4. the parameter values (except for those that are of interest).

The algorithm will then find the shortest route from the known to the unknown values, once it has done this it will output a list of the constraints which are on the path, all of which must be satisfied. This can then be put into a symbolic equation solver to produce an equation which gives each of the unknowns as a function of the known signals and the known parameters.

4.5 Software

In this section an outline of the software which has been developed will be given. This will look at the overall way in which various software modules are used to break down the problem of automatically generating a specific fault detection and diagnosis application for a particular system. The flow of information between these will also be examined.

The software is divided into two areas. There is some software which is used off-line. This is used to analyse the system to be monitored and generate the actual fault detection and diagnosis software for the system. This automatically generated software then forms the on-line software which can be copied to the computer which

is to perform the actual detection and diagnosis, and is then compiled ready for use. These are now examined in a little more detail.

4.5.1 Off Line Software

Figure 4.9 gives an overview of the off-line software.

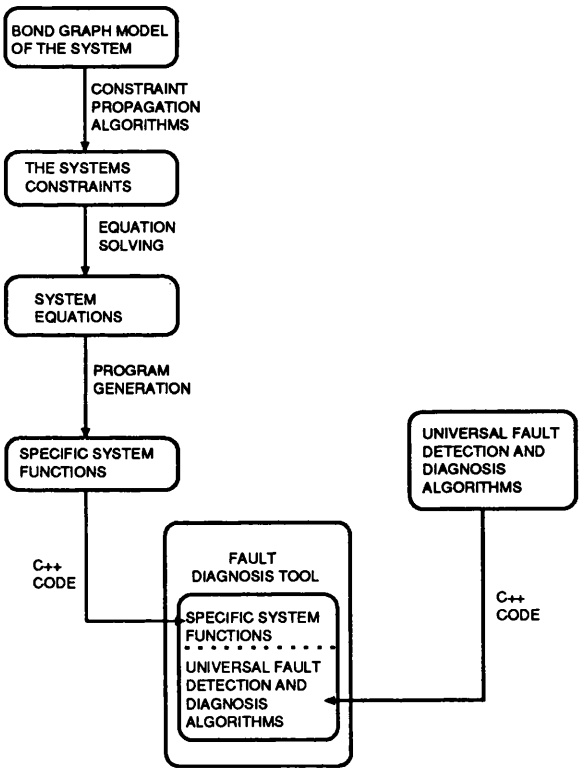


Figure 4.9: Overview of the off line software.

This begins with a bond graph model of the system, which is the representation of the system as given by the system modeller/user. The constraint propagation algorithms, described earlier, are then used to find which component relations and sensor measurements are needed to identify various system signals and component parameters. This information is then passed to a symbolic equation manipulator which repeatedly takes each set of relations, solves them (if this is possible) and stores the result. These system equations are then incorporated into C++ syntax functions which are combined with the universal fault detection and diagnosis algorithms to

yield a complete fault analysis tool specific to the system.

4.5.2 On Line Software

The overview of the on-line software is shown in figure 4.10. From the system that is being monitored, measurements are taken at regular intervals from different sensors attached to the system. This data is then passed to the fault analysis tool.

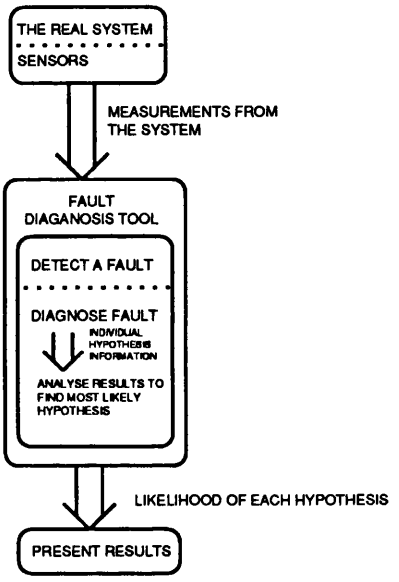


Figure 4.10: Overview of the on line software.

The on-line software firstly waits for a fault to be detected and then fault hypotheses are tested to see which best fit the new behaviour of the system. The results of these individual hypothesis tests are then used, to look at sets and subsets of multiple fault hypotheses to determine which are mostly likely to be correct. The results are then presented to the operator.

Summary

The software is divided into two sections, off-line software and on-line software. The off-line software automatically generates the code for the on-line software. In the on-line situation, the system is monitored, waiting for a fault to be detected. After

this, hypotheses are tested to find how well they fit the new behaviour of the system. The results of this are then extrapolated to the other hypotheses and presented to the user.

4.6 Prefiltering Measured Signals

The effects of noise (see figure 3.19.D) on the measurements from the system which is being monitored can severely degrade the accuracy of both the fault detection and diagnosis algorithms. There are a number of steps which will be taken to reduce these effects, most of which will be discussed in chapters 5 and 6. Here an initial step in reducing the effects of noise will be discussed, this works by attempting to reduce the noise which is present in measured values received from the sensors attached to the system. Alternatively, a separate prefilter could be used.

Firstly what are the requirements? If multiple time period models of a system are being employed, then to use it there must be data from more than one time period available. There must be the current value from the sensor, and a number of previous values. This is required for each sensor on the system. So suppose an n time period model were being used, the last n values from each sensor would be needed, and it is desired to reduce the presence of noise in these values.

This is done in two steps, first of all a note is made of the fact that it is not actually necessary to use the current measurements. To explain, suppose samples are taken every $\frac{1}{20}^{th}$ of a second. These measurements could be stored in a first in first out buffer of, for example, ten samples long. It is possible to then regard the 'current' value from the sensor as the value coming out of the buffer. This will mean that the model is in fact $\frac{1}{2}$ a second behind the real situation of the system. The advantage of this is that there is a value from the sensor which can be regarded as the current value, and there are also ten readings from the sensor which can be regarded

as the future values which will come from the sensor. It is now possible to look at the past values from the sensor, look at the future values from the sensor and look at its current value, decide how the values are behaving in general and adjust the values of sampled data so that it conforms to this general behaviour, and thus reduce the noise in the measured signals.

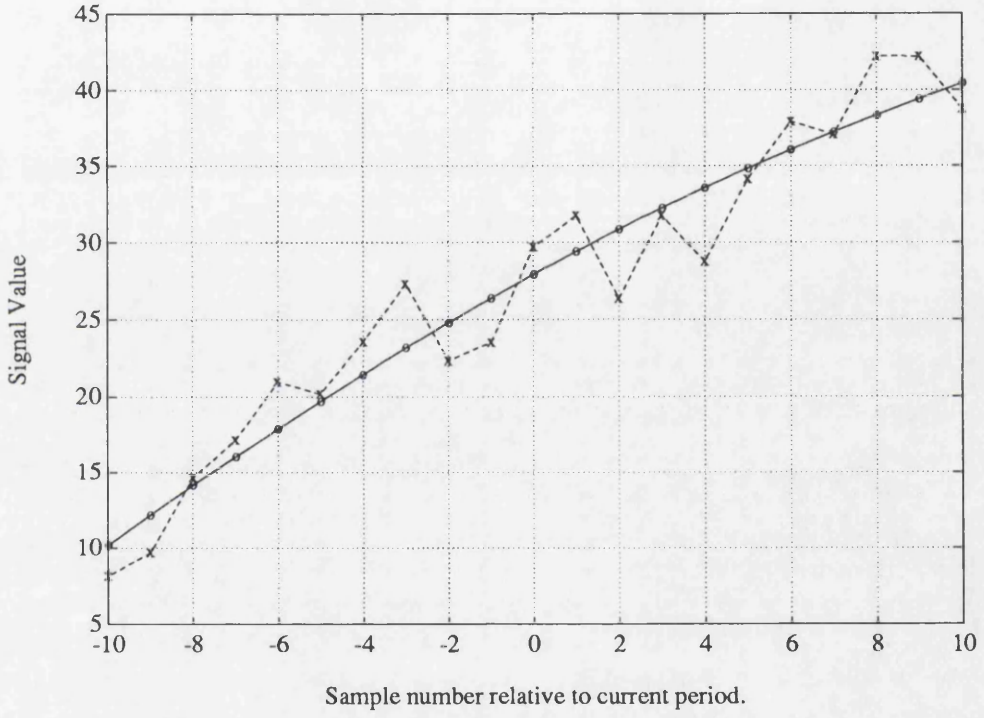


Figure 4.11: Fitting a parabola to a set of data.

To do this, the current value from the sensor is taken along with a number of future readings and the same number of past readings, then a 'best fit' parabola is made to this series of data points[73], and instead of using the actual values of the data, the corresponding values from the parabola are used. A typical situation is shown in figure 4.11. The dashed line represents the value measured by a sensor, the '×'s are the actual sample values. The full line is a best fit parabola to the data. The 'o's are the corresponding values on the best fit parabola. The sum of the square of the error between each measurement and each corresponding value on the parabola

has been minimised. This parabola is the least squares best fit parabola. It can be clearly seen that in this case the quality of the data will be increased by using the points from this parabola rather than the actual data points.

If a three time period model of the system is being used, then the data from the parabola (marked 'o') from the times labelled 0, -1 and -2 would be used. That is the current data value and the two previous ones to it.'

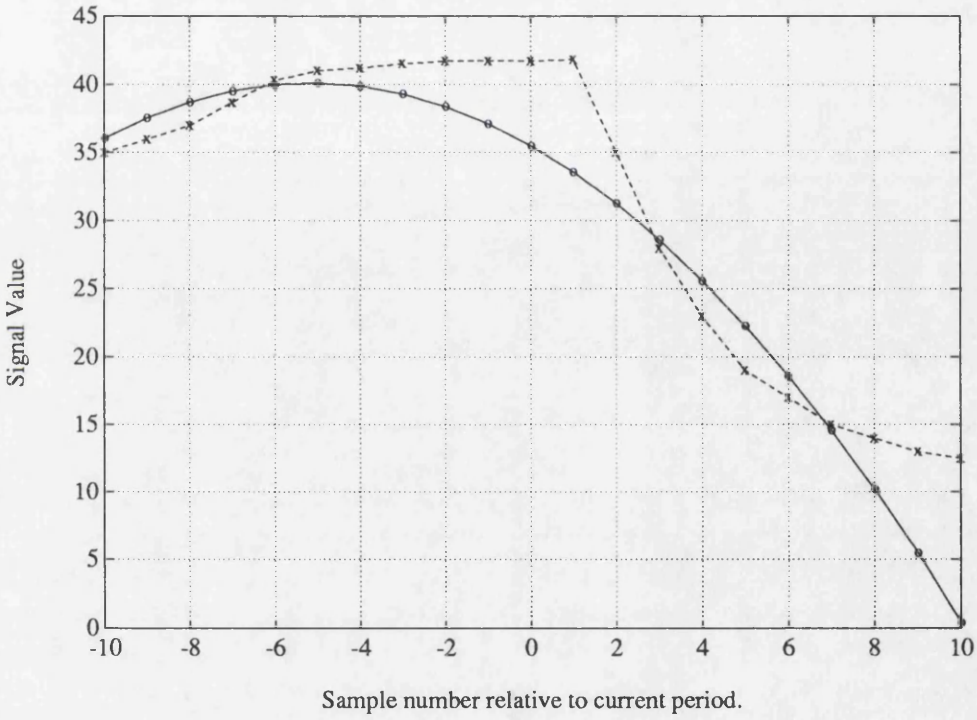


Figure 4.12: A poorly fitting parabola.

Figure 4.12 shows another situation where again the actual data is the dashed line and the best fit parabola is the full line. Here there appears to be little noise on the actual data, but the parabola fit is poor, especially around the centre of graph which is the area of most interest. The error between the points at $T = 0$ is approximately 6 units. There are number of possible ways in which this can be improved. Firstly, if it is expected that a measured signal will have quite a few sudden changes in value, then it may be more appropriate to use a smaller set of data to fit the parabola to. In

figure 4.13 same data is shown, but this time the parabola is only fitted to ±5 units of $T = 0$. The error at $T = 0$ is now approximately 4 units. This has reduced the error but the parabola is still a poor fit.

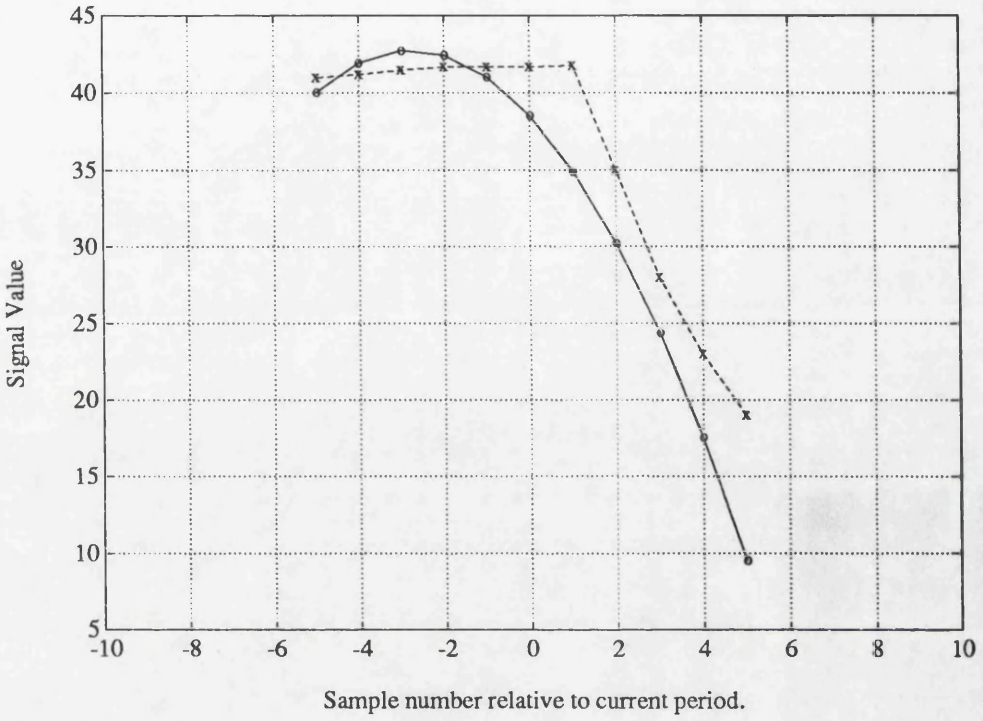


Figure 4.13: Using a smaller data set for fitting.

A second option is to put a limit around the parabola, and any measured data points which are outside this limit are assumed to be more correct than the parabola. This is shown in figures 4.14 and 4.15. Figure 4.14 shows the same situation but with a 5% band placed around the parabola (shown by the two dotted lines). If the measured data point is inside this limit then the point from the parabola is taken, otherwise the actual measured point will be taken. The result is shown in figure 4.15, the dotted line is the actual data and the full line shows the values that would be used in the analysis algorithms. This full line is simply made up from parabola points (-10 to -4,3 and 7) and actual data points (-3 to 2,4 to 6 and 8 to 10). The error at $T = 0$ is of course zero, but it is likely that there will be more noise present now

that the actual measured data values are being used.

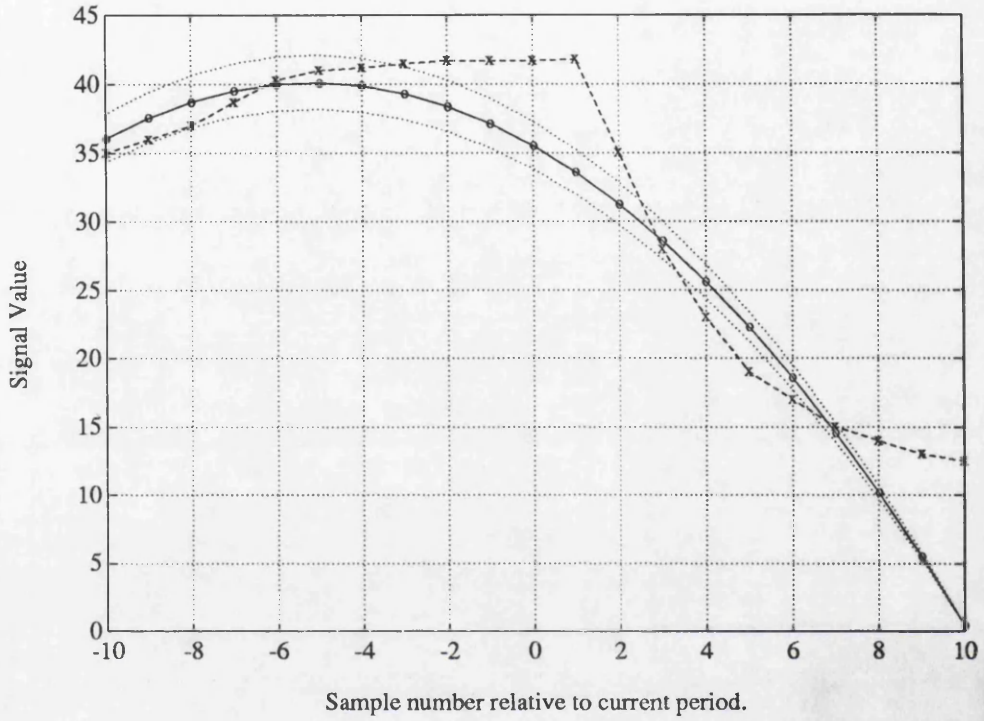


Figure 4.14: A 5% accuracy limit around the parabola.

The number of data points used for fitting should depend on the expect characteristics of the signal to be measured, i.e. for high noise, no sudden changes situations, use many data points to fit the parabola. For low noise and sudden changes in measurements, if possible use the unfiltered data or if some noise is present use just a small number of data points to fit the parabola. High noise and sudden changes in signal levels is the hardest case to deal with. If the measurements from a sensor have these characteristics, then it may be better not to use that sensor at all, but instead find another nearby location and take a measurement from there.

There should always be a limit put on the accuracy of the parabola. If the expected noise content of signal is 2%, then a similar limit should be put on the parabola, if the actual data is outside this limit then it is probable that the actual data is more correct than the parabola and this value should be used in preference. This will

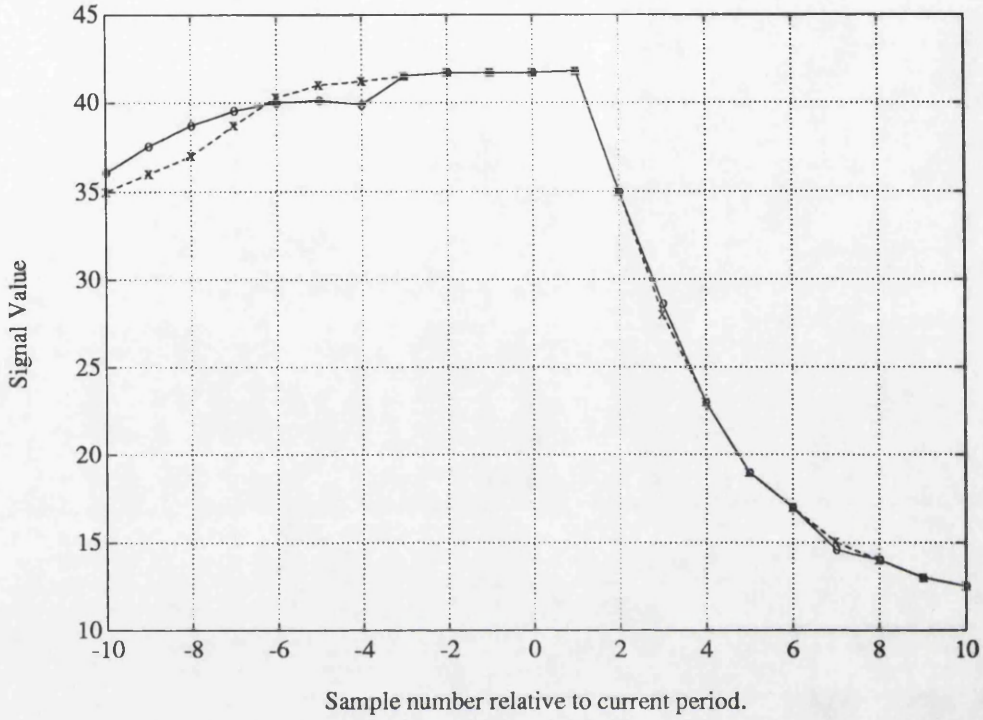


Figure 4.15: The resulting data for use in the analysis algorithms.

prevent sudden, possibly unexpected changes in the characteristics of a measured signal from going unnoticed by being filtered out.

Chapter 5

Fault Detection in Dynamic Systems.

The objective of fault detection[1] is quite simply to be able to tell the difference between a system which is working correctly, and one which is misbehaving. By misbehaving it is meant that it is not performing to its specifications, within some predefined tolerance. *"A fault is to be understood as a nonpermitted deviation of a characteristic property which leads to the inability to fulfill the intended purpose."* [41]

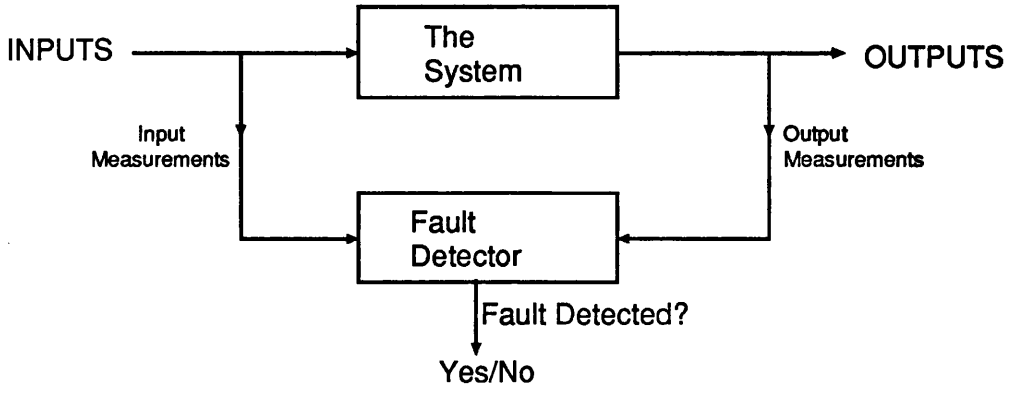


Figure 5.1: The method for detecting faults.

The general appearance of fault detection methods is shown in figure 5.1. [12]

[1]The separate issue of fault diagnosis is treated in chapter 6.

[10] [8] [11] [36] [68] [19] [41] [63] [64]. The inputs to the system and its outputs are measured, and from these checks are made as to whether the system is behaving correctly. There are a number of different ways for performing the fault detection, depending on the type of system, and the required sensitivity of the fault detector [1] [46] [36] [8] [21] [71] [52].

The detection method is a model based approach using structure, function[8] [71] and causality [45], and so in the box titled 'Fault Detector' a comparison will be made between these measured signals and the corresponding signals from a model of how the system should be behaving. Discrepancies between the expected behaviour and the actual behaviour which indicate the presence of a fault will be identified.

Although the concept of fault detection is straight forward, there are a number of issues which must be considered and catered for to produce a usable method for detecting faults in dynamic systems. The objectives for the fault detection algorithm are:-

1. The fault detection algorithm should be made as general as possible. It is desirable that the fault detection algorithm is usable on as wide a range of systems as possible with the minimum amount of modification, and preferably none.

2. The fault detection system should be robust with respect to false alarms. In general there will be noise in the measurements taken from the system which is being monitored, also the parameters in the model of the system will not be exactly the same as the real systems parameters, but will be just estimates of them. In the face of this uncertainty the potential for false alarms is great. Noisy measurements and model uncertainty need to be catered for. [13] [14] [19] [63]

3. The fault detection system should be as sensitive as possible and detect small

changes in the system. This is in opposition to 2, above. It is desirable for the fault detection algorithm to detect small faults, but obviously this is limited by the amount of uncertainty in the model parameters and the system measurements. The system must therefore not be so resistant to false alarms that genuine faults of a significant size are ignored.

Compromise is required between all three of the above. If a fault detection method for one specific system was being designing, then 2 and 3 could be tailored for that system. Objectives 2 and 3, the balance between sensitivity and false alarms, are clearly in conflict. These areas will now be looked at in more detail and some solutions for problems which arise will be presented.

Firstly, a fault is defined, and the causes of false alarms are identified. How to monitor a system will be looked at, that is, how the model is made to follow the current state of the system, so that the model is a reflection of the real system. This will involve estimating states, and will require some filtering to reduce the effects of noise. Our model will never be 100% accurate, and so it is necessary to take into account the effects of modelling errors in the monitoring system. Finally, one method for comparing the outputs from the real system with the outputs from the model is looked at, and this comparison is used to determine whether a fault is present.

5.1 Definition of a Fault.

The definitions of a fault that is used here is given in the following two statements. The first one identifies three different types of fault, and the second one indicates the difference between a fault in a system and a fault in a component.

1. *A fault is defined as being the change in the parameter of a component, or a change in the structure of the system or a fundamental change in the behavioural*

characteristics of a component.

2. *If more than one component has failed then this is a multiple fault i.e. one fault in the system which is made up of the failure of more than one component.*

It should be noted that there is a great difference between detecting a fault and diagnosing it. Detecting means realising that something is wrong and diagnosing involves finding what has gone wrong.

Statement 1 defines what types of faults are to be **detected.** Unfortunately all of these types of faults cannot be **diagnosed** with the methods used here, as will be discussed fully in chapter 6. Whenever a fault occurs, it is required to detect a fault as being present in the system, although sometimes it will not be possible to say exactly what is wrong.

Multiple faults can be caused by a chain reaction effect i.e. one component fails, which quickly causes another one to fail and so, or they may be coincidental, two or more components just happen to fail at round about the same time. In either case, the ability to detect these faults and if possible diagnose them is required.

When a fault occurs in the system which is being monitored, it will no longer behave in the same way that the model of the system does. This fact is used to detect the occurrence of faults. There are, however, some other ways in which behavioural differences can arise.

The actual cause of discrepancies between the real system and the model of the system could be due to any one of the following.

1. An unmodelled structural part of the system which has failed.
2. Unmodelled characteristics of the system which were thought not to be significant turn out to be significant.
3. Poor estimates of the parameters in the system model.

4. Excessive noise in the system or the measurements taken from the system.

5. Estimates of the system's states are poor.

6. The parameter of one or more components has changed.

7. The fundamental characteristics of a component have changed.

Only the occurrence of 1, 6 and 7 above are to be detected, as these are the only ones which are caused by a failure within the system. Although in the case of 1 and 7 diagnosing the cause of the fault will not be possible, as changes in the structure of the system or fundamental characteristics are very difficult to diagnose[2] and are not catered for here. It is desired that 2, 3, 4 or 5 do not cause a fault to be detected as these are not caused by a fault in the system, and are thus false alarms.

In the case of 2 & 3, it may be possible to go back and remodel part of the system or improve the parameter estimates to prevent these from causing a fault to be indicated, which will obviously take some effort. Alternatively, the detection algorithm could be informed that the system is not well known, this will decrease the sensitivity of the detection algorithm but will help prevent false alarms due to modelling errors.

In the case of 4, if the measurements contain noise then the detection algorithm should be able to reduce its effects and prevent discrepancies between the systems outputs and the model's outputs due to noise from causing a fault to be indicated. To do this without excessively reducing the sensitivity of the detection algorithm, the detection algorithm needs to be informed of the expected noise content of measured signals.

When monitoring a system first starts, the estimates of the system's states are likely to be poor, but these estimates should improve with time. It is necessary to be

[2]unless prior knowledge about the physical structure of the plant is used.

aware of how accurate the state estimates are likely to be when a fault is about to be indicated. If the discrepancy between the actual system and the system's model is within a limit imposed by the current accuracy of the state estimates, then the discrepancy can be accounted for as being due to poor state estimates, rather than the presence of a fault. Doing this will reduce the chances of 5 from above causing a false alarm.

5.2 Monitoring a System.

The first thing that must be done before a fault in a system can be detected is to find out what state the system is in, and track the systems behaviour using its model [41] [19]. This involves using the measurements of its inputs and outputs to find the value of the states in the system.

The fact that the model's parameters (and possibly its structure) will not be 100% accurate must be taken into account, and that the measured signals will contain noise. This means that the state of the model will only approximately reflect the true state of the system.

The idea is that using the model and the signals measured from the system over a period of time, it is possible to predict how the system should be behaving, providing that the system is not faulty. The measured outputs are assumed to be correct given the measured inputs. The model and the measured inputs are then used to generate expected values for the outputs. The expected outputs and the actual systems outputs can then be compared. If there are significant discrepancies then the conclusion that the system is faulty is made. While this is being done the outputs must continue to be measured to correct any errors in the model's states which may arise due to the approximate nature of the model. To correctly monitor the system, the states of the system need to be estimated, and it is important to be aware that the parameters

are inaccurate and the measurements contain noise [63]. These areas will now be examined in turn.

5.3 Initial Estimate of States

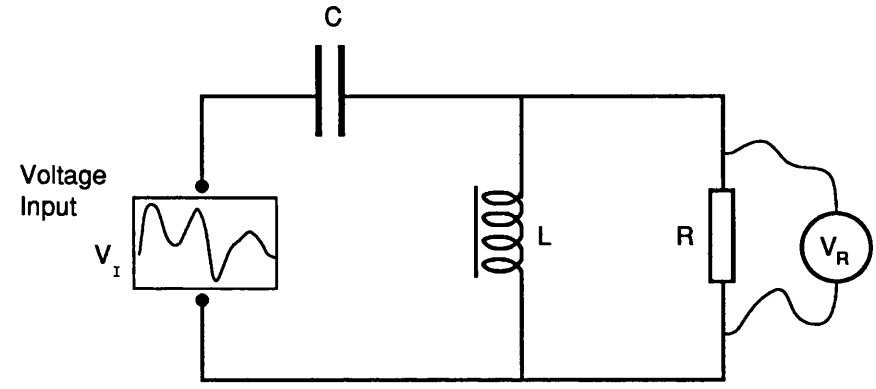
Here a state estimation method will be presented based upon propagating signals from the system's inputs and outputs to its states and hence finding their value. For the purposes of clarity, it is assumed that the model parameters are completely accurate although the measurements do contain noise. The way in which states are estimated will be illustrated with an example.

Figure 5.2.A show a simple electrical circuit. V_I is the voltage input to the system which is being measured and which varies with time. V_R is the voltage across the resistor (and inductor) which is also being measured. From these two measured voltage signals it is required to estimate the two states in the circuit. Figure 5.2.B shows a bond graph representation of the system and figure 5.2.C shows the equivalent block diagram. The states are the charge on the capacitor and the flux of the inductor, these can be thought of as being the signals V_C and i_L on figure 5.2.C respectively.

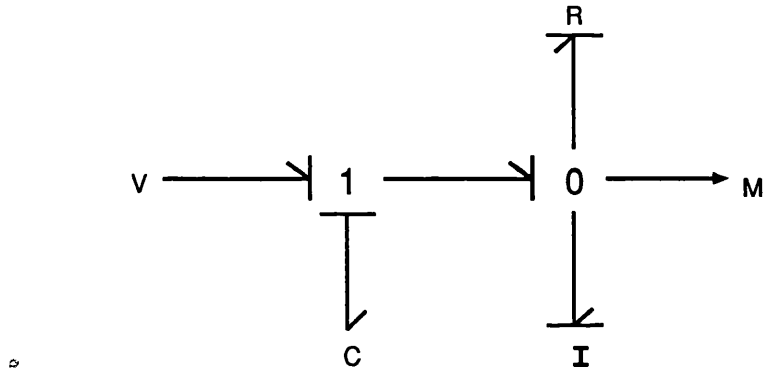
Figure 5.3 shows measurements of the applied voltage and the resulting voltage across the resistor, these measurement have approximately a 10% noise to signal ratio. The components had the parameters shown in table 5.1 and samples were taken 40 times per second. These measurements are firstly used directly without the prefilter described at the end of chapter 4, and one method of estimating the states from these measured signals will be looked at.

Component	Parameter
R	20 Ω
C	50 mF
L	5 H

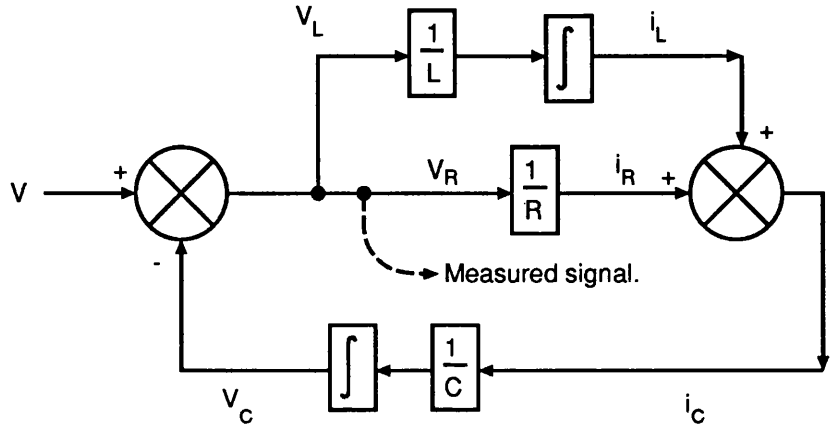
Table 5.1: The component's parameters.



A: Circuit Diagram.



B: Bond Graph.



C: Block Diagram.

Figure 5.2: Estimating the states in a simple electrical circuit.

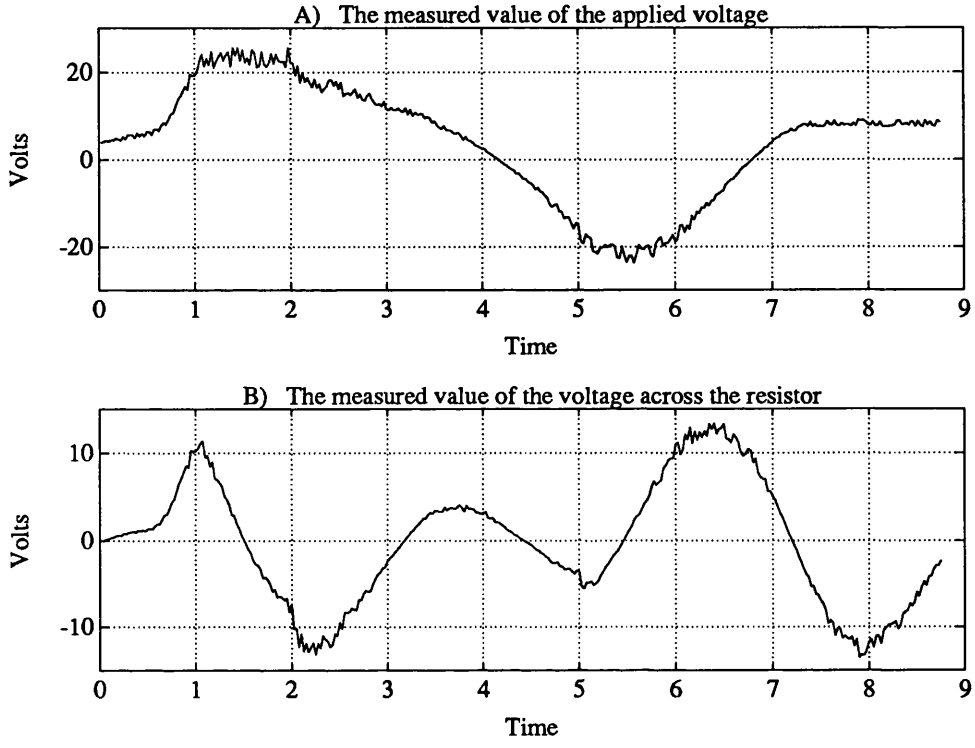


Figure 5.3: The applied voltage and the voltage across the resistor.

Firstly, just a two step model of the system will be used, together with two observations at a time to find values of the two states. The results of this are shown in figure 5.4. 5.4.A shows the estimate of the state of the capacitor (V_c in figure 5.2.C). The full line is the true value of the state, and the dotted line is the estimate obtained from using the measured signals. The estimate does contain noise, but the estimate always remains 'close' to correct value.

Figure 5.4.B shows the estimate of the state of the inductor, (i_L in figure 5.2.C). This estimate is very poor because of the need to differentiate the capacitors state in order to calculate it (as its initial state is not known). The estimated value now only vaguely appears similar to the true value. Clearly, using this kind of estimate for a state will produce vary large errors if it were used in further calculations. To improve the situation a Kalman filter like approach is taken to strike a balance between the noisy but essentially correct value for a state produced by propagating backwards

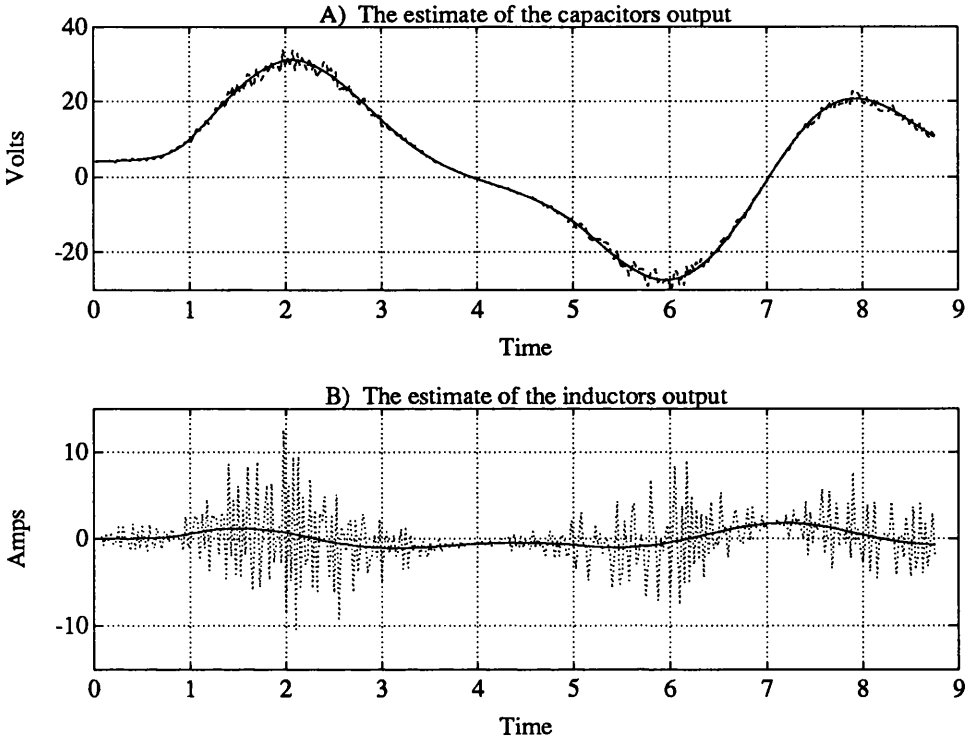


Figure 5.4: State estimates without any form filtering.

from the outputs, and a relatively noise free but not quite correct value for a state produced by propagating forwards from the inputs and using the previous values of state estimates [79] [63] [67].

Figure 5.5.A and B show a second set of results but this time a filtering technique is employed. At the beginning the estimates are still poor, but they quickly improve and become very close to the true values of the states. The estimate of the inductors state takes a longer period of time to closely approximate the true value than for the capacitor, but it can be seen that the effects of the noise are drastically reduced.

The filtering was done in the following way. There are two ways to calculate values for the states, one is to use a number of observations and the same number of system models joined together, and then essentially propagate the signals from the observations throughout these models and so find values of the states for the current time period. This is what was done for figure 5.4. The problem with this

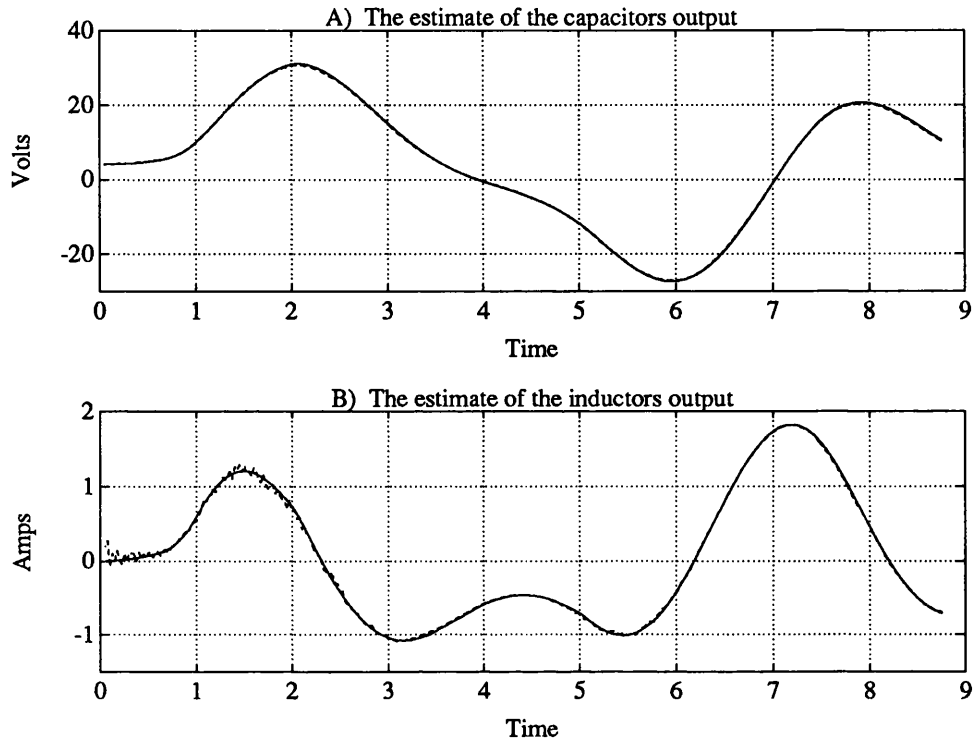


Figure 5.5: State estimates with filtering.

is that noise is amplified when signals are propagated backwards through a dynamic component, although the value for the state will be correct, but with the addition of a large amount of noise. The second method is to use a 1 time period model of the system and use only the system inputs and the previous value of the states, but not the system outputs. The inputs and previous state values are then propagated throughout this 1 time period model and the value of the states and the outputs for the current time period are found. The values for the outputs calculated here will be used later, these values are estimates for the outputs based on the inputs and state estimates. The problem with this is that the initial value of the states need to be known, also if the initial states are known but are not accurately known, then the current values for the states will also be inaccurate, however noise will not be amplified.

The filter method used, combines both of these, and it is based on the following

line of reasoning. At the start of the monitoring process it is not known what the value of the states may be, so a multi-time period model of the system is used to find values for the states. These values will contain high levels of noise, but at least now we have some idea, although it may be poor, of what the states are. For the next time period the same thing can be done, but the state calculated at the previous time step and a one step model of the system can be used to calculate another set of values for the states during the current time period. This second set of values will not suffer from amplified noise, but will be inaccurate because the initial values for the state were inaccurate.

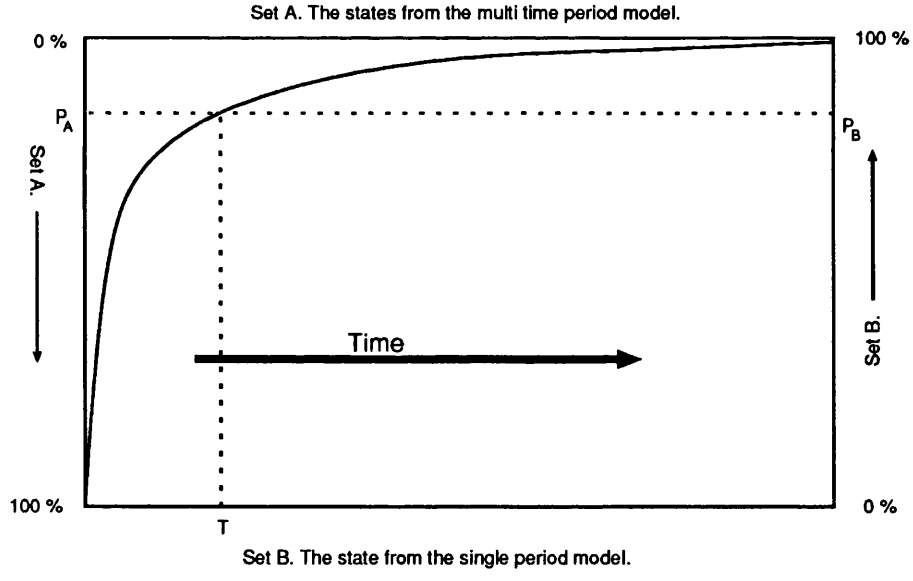


Figure 5.6: The ratio between the 2 set of values for the states.

For each time period there are now two sets of the values for the states, one from a multi-time period model, and one from a single time period model using the calculated states from the previous time step. What is done now is to combine these two sets of values for the states to produce a 'best guess' of what the states are for the current period. These states are combined in the following way. The value for state A from the first set of values and the value for state A from the second set are taken, and basically a value which is between these is chosen, but the way in which this

value is chosen is very important. At the start of the monitoring process values which are close to the set of values obtained from the multi-time period model are chosen. As time progresses values which are closer to the values from the single time period model are taken. This third set of values for the states will be used in the single time period model as the initial value of the states in the following time interval.

Figure 5.6 shows the method which is used for calculating the 'best guess' for the state values. The value of the state for the current time period is given by $state = P_A \times A + P_B \times B$, where A is the value of a state from a multi time period model, B is the value of the same state from the single time period model, and P_A and P_B are given by the graph in figure 5.6. Note $P_A + P_B = 100\%$. A diagram of the scheme of all of this is shown in figure 5.7. The 'current state estimates' are saved and used during the next time period as the 'state estimates from the pervious time period'. The function shown in figure 5.6 which was used for this example was, $P_A = \frac{1}{1+10\frac{T}{N}}$, where T is the number of time periods that have passed since monitoring began, and N is the expected noise content of the measured signal, in this case 0.10 as the noise to signal ratio is approximately 10%. $P_B = 1 - P_A$.

The estimated output (mentioned earlier) in the R-L-C circuit example is shown in figure 5.8. Comparing this with figure 5.3.B The measured output and estimated output are seen to be essentially the same, the only difference is caused by noise.

5.3.1 Initial Parameter Accuracy.

In any model of any system there will always be some approximations made when it is produced. This may take the form of unmodeled dynamics and components which are considered insignificant when compared to the rest of the system [45] [33] [23]. There will also be some error in the parameters of the components that are modelled. Obviously the task of the modeller is to reduce these approximations to a minimum,

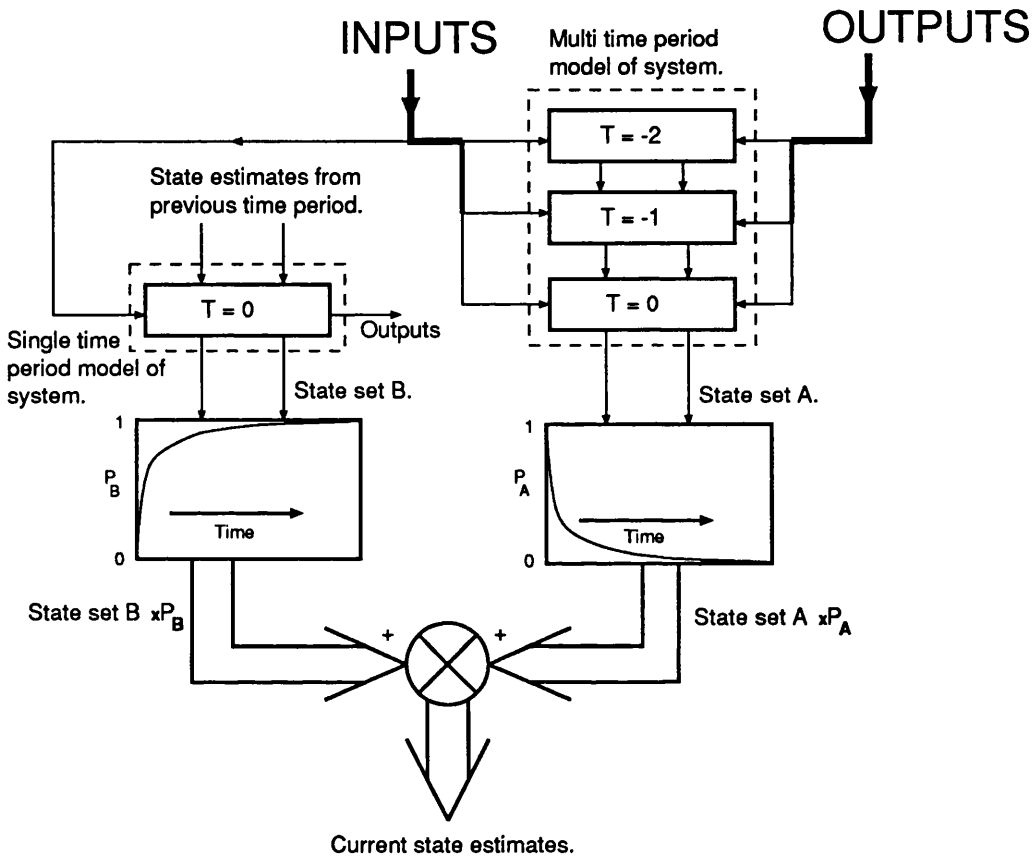


Figure 5.7: The scheme used for estimating the states.

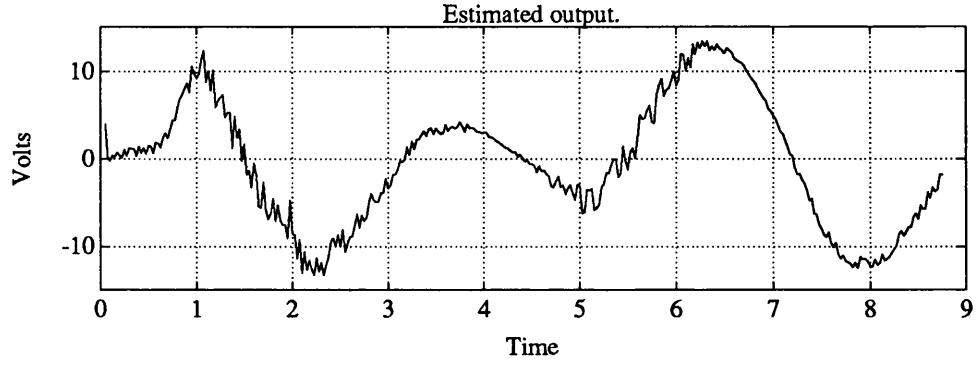


Figure 5.8: The estimate of the systems output based on the inputs and state estimates.

but they will always be present. If a system is to be monitored over a long period of time it is necessary to realise that the model is not perfect. This is so that a false alarm is not caused simply by a small modelling error being amplified over a long period of time thus appearing to be a large discrepancy with the real system.

In the R-C-L circuit considered earlier, lets suppose the system has the parameters shown in table 5.1, but the model is incorrect, the inductor in the model has L = 6 rather than 5. This represents a discrepancy between the system and the model, but this should not cause a fault to be detected as it is assumed that this is a modelling error and not a fault.

Figure 5.9 shows the results if it is assumed that the model is 100% correct. The full lines are the actual values (without noise), and the dotted lines are the estimates. As time progresses the noise in the state estimates reduces, but there is an offset compared to the actual value of the states, also the estimated output is offset from the measured output. This can be enough to detect a fault because the difference between the actual and estimated outputs cannot be accounted for by just the presence of noise: something else must also be present i.e. a fault.

This is coped with by informing the state estimating algorithm from the previous section that the model is not perfect and it is given an accuracy index n of between 1 and 0. 1 meaning the model is 100% correct, and 0 meaning the model is completely wrong. Then, rather than as before (see figure 5.6) where the values for the state estimates eventually become 100% of the single model states, a limit is put on this of $n \times 100\%$. e.g. if the model is assumed to be 95% correct, then $n = 0.95$ and the state estimates will eventually consist of 95% of the values from the single time period model and 5% from the multi time period model. Now $P_A = n \times \frac{1}{1+10\frac{T}{N}}$ and $P_B = 1 - P_A$. Now some fraction of the states derived from the outputs will always be present in the estimates of the states. If the states start to drift away from their

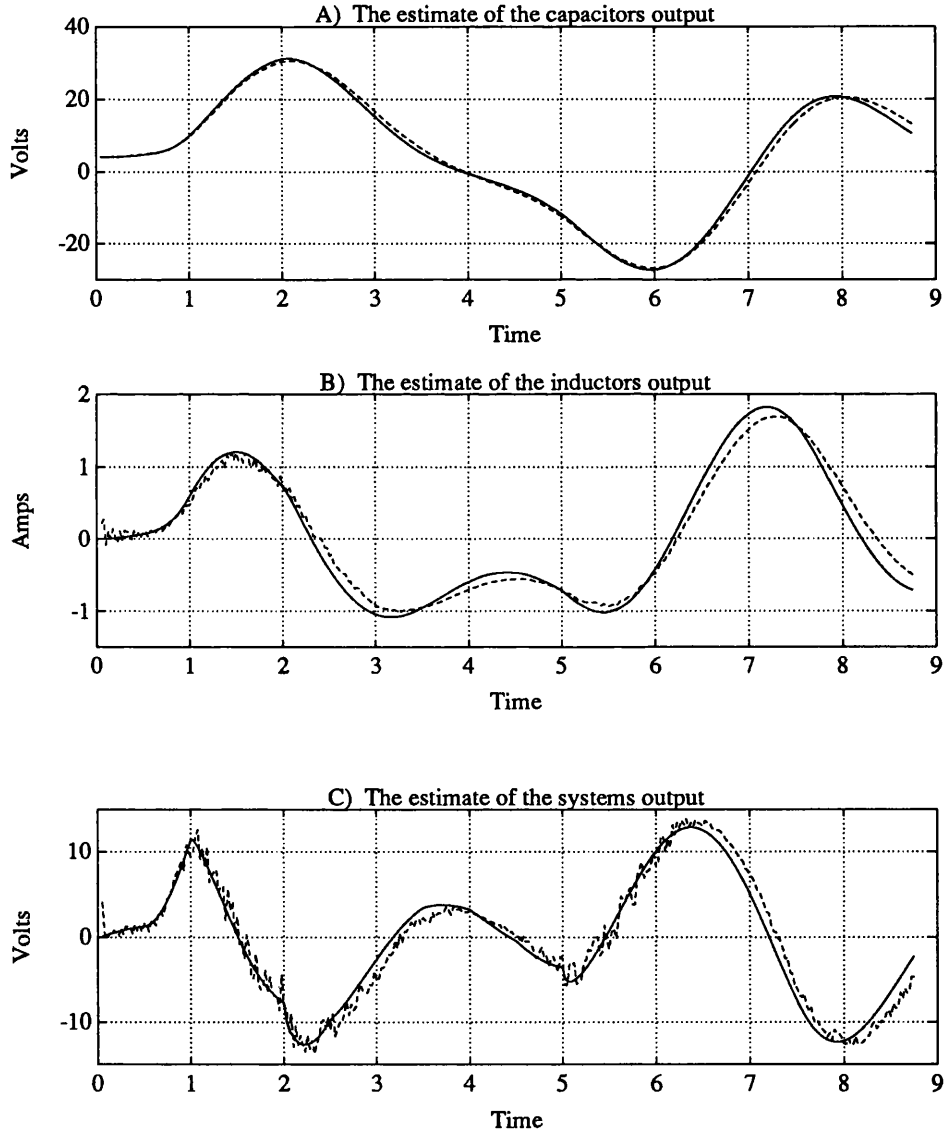


Figure 5.9: The effects of not taking into account modelling errors.

correct values because of a modelling error, this will help to move them back to their correct values. This does mean however that slightly more noise will appear on the state estimates.

Figure 5.10 is the result when the accuracy index n is set to 0.93 for the R-L-C circuit. This means an assumption that the model is about 93% accurate has been made. These graphs can be compared directly with those in figure 5.9, it can be seen that the noise content of both the state estimates and the output estimate has increased (although figure 5.10.B is still vastly better than the unfiltered version in figure 5.4.B), the offset effect has been removed and the discrepancies in the estimated and measured output can be accounted to noise alone.

It should be noted that in general the offset may not necessarily be removed from the state estimates, this depends upon which parameters in the model have the largest error. However it will always be possible to remove the offset from the estimate of the output. This is because the states are a function of the inputs, outputs and parameters, and the outputs are a function of the inputs, the states and the parameters. This means that any modelling errors or errors in the parameter estimates will cause state estimates to be incorrect. However, when the values for the states are propagated to the outputs the same modelling errors will be passed through, but in the opposite direction. This will cause modelling errors to be cancelled out when the output estimates are made. This is important because if output errors due to known model inaccuracies can be reduced, then false alarms due to this are also reduced.

5.4 Detecting the Occurrence of a Fault.

In this section a decision is made as to whether or not there is a fault present in the system. As described in the previous sections the values for the states can be

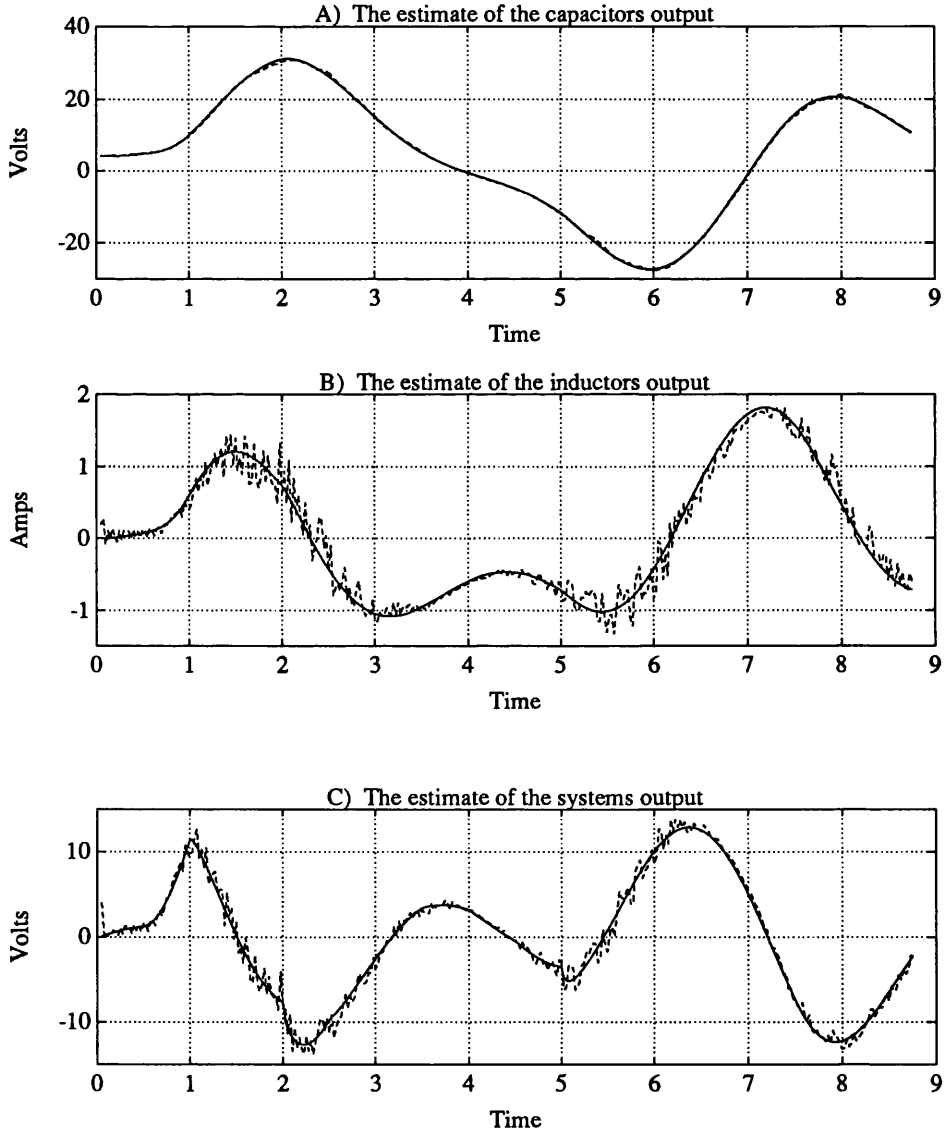


Figure 5.10: Taking modelling errors into account reduces false alarms.

estimated and these can be used to produce an estimate for the outputs from the system. Modelling errors can also be inhibited from producing significant errors in the output estimates. What is done here is to look at these estimates of the system's outputs and look at the actual measurements from the real system and compare them. If they are significantly different, then a fault is indicated as having been detected.

Before this can be done, significant differences need to be defined and a decision about how to cope with very small signals or very large signals needs to be made, as these can present problems.

Firstly it is desirable to be able to use this for a wide range of system, where a system's outputs may typically be of the order of 10^{-3} and another system's outputs may be around 10^{6} for example. The detection method should work equally well when handling either of these sizes of numbers. An error index is calculated which indicates how well the measured and estimated outputs match each other. The error index will have a value of zero if the actual and estimated outputs are identical, a value of 1 or greater indicates a complete mismatch between the outputs and their estimates. For a system which has noise/signal ratio of 0.1 (i.e. 10% noise), then typically this error index is expected to fluctuate between about 0.08 and 0.12.

To calculate such an error index first of all the measured outputs and actual outputs are compared by finding the absolute percentage error between these values. This will allow the use of a standard range of numbers from 0% to 100% or there abouts. The results of doing this for the R-C-L circuit are shown in figure 5.11. The measured output can be seen in figure 5.3.B and the estimated output in figure 5.10.C. Figure 5.11 is given by $\frac{Measured\ output - Estimated\ output}{Measured\ output} \times 100$.

The first thing that is noticeable about the results in figure 5.11 is the five peaks which indicate a very large percentage error. The position of these corresponds to the times when the measured output crosses zero. At these positions the true value

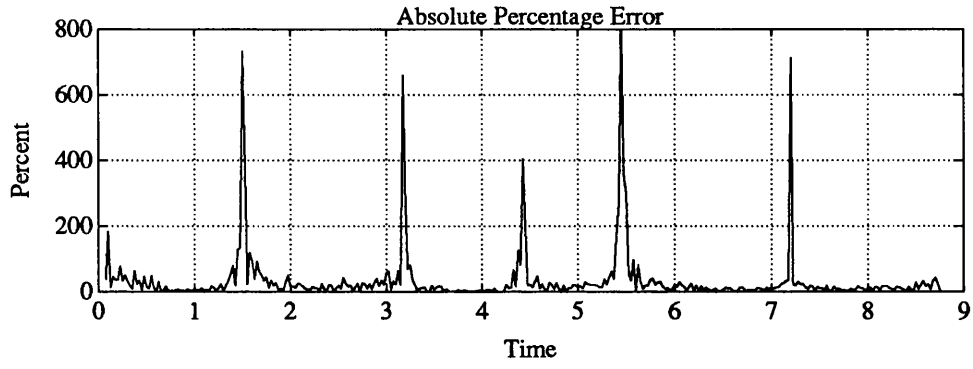


Figure 5.11: The difference between the measured and estimated output.

for the output is small, but the noise levels are high in comparison i.e. the true signal is being swamped by the noise. What must be realised here is that values for the estimated output and the measured output are virtually meaningless as they consist mainly of noise. To get around this problem a limit is introduced on the magnitude of each output, when the value of the output is below this limit then it is omitted from the error index calculation. Also, when the value for an output is above the limit but is still close to it, its significance in the error index calculation is reduced. The error index for each output at any one period of time is given by the following equation. It is based on the same lines as the recursive least squares algorithm used in parameter estimation, except here the 'parameter' is the output error ratio [73].

$$Error\ Index = \frac{... + E_3.F_2.F_1.F_0.W_3 + E_2.F_1.F_0.W_2 + E_1.F_0.W_1 + E_0.W_0}{... + O_3.F_2.F_1.F_0.W_3 + O_2.F_1.F_0.W_2 + O_1.F_0.W_1 + O_0.W_0}$$

Where

n the subscript for each variable is the number of time periods backwards from the current time, i.e. $n = 0$ is the current time, $n = 1$ is the previous time step etc.

E_n is the absolute size of the error between the measured and estimated value of the output. This corresponds to figure 5.11.

O_n the absolute value of the measured output.

W_n is the weight associated with the magnitude of the output.

F_n is a forgetting factor calculated for each time period n.

If the output is below the limit described above, then W_n is zero. If the output is less than $4 \times limit$, then $W_n = 1 - \frac{4 \times limit - output}{4 \times limit}$, i.e. if the output is just above the limit then W_n is slightly larger than zero. As the output moves away from the limit, W_n increase to 1. If $output > 4 \times limit$ then $W_n = 1$.

A value of 0.98 is used for F when O_n is greater than $4 \times limit$, $F = 1$ when O_n is less than $limit$, and $F = 1 - 0.02 \times W_n$ when O_n is greater than $limit$ but less than $4 \times limit$. This means that when the measurement of the output is small none of the previous results are forgotten (since $F_n = 1$), and the current result is not included in the error index (since $W_n = 0$). If the measurement is a good high signal then $F_n = 0.98$ and $W_n = 1$, i.e. a little is forgotten about the previous results and the current results are used fully.

The error index can be thought of as the percentage error between the measured and estimated output based on the current and previous time periods, and ignoring small measurements from the output when noise levels can be very high in proportion to the signal.

If the system has more than one output then the error indices for each output are simply added together and divided by the number of outputs. This is then an approximate measure of the average percentage error of the outputs.

Figure 5.12 shows the error index (the full line) for the R-L-C example. Here the limit below which measurements are ignored is 1.5 volts. This figure shows the contrast to the actual percentage error for each time period shown in figure 5.11. The peaks have disappeared and the value of the index remains fairly constant at between about 10 and 18 %. The dotted line is the limit within which the error index must

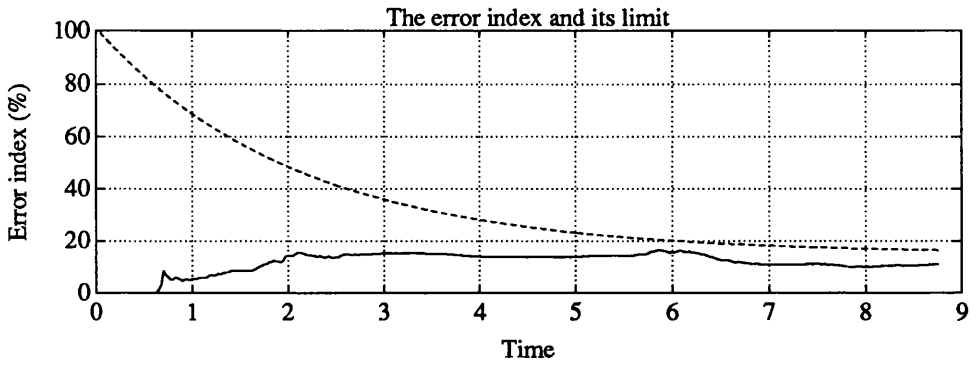


Figure 5.12: The error index and the error index limit.

stay if there is no fault in the system. If a fault develops the error index will increase and go above this dotted line, and a fault is then detected. The error index is allowed to go above this limit briefly. This is because step changes at the inputs can produce changes in the expected output at slightly different times to the actual outputs due to exactly when samples from the system are taken. This produces step increases in the error index, but is not caused by the presence of a fault. Therefore a fault is not detected if the error index is larger than the error index limit, and the error index is decreasing. This prevents false alarms due to step changes at the system inputs.

The error index limit starts at 100% at the start of the monitoring process and as time increases it reduces to $1\frac{1}{2}\times$ the expected noise/signal ratio on the measurements (expected noise for the R-L-C circuit was 10%). This is because at the start of the monitoring process the states are not accurate so the estimated output will not be accurate and large errors can be produced, but as time continues the state estimates improve as do the estimated outputs and therefore the error index reduces unless a fault is present. Assuming that there is some measurement noise, then even if a perfect estimate of the outputs without any noise is obtained, the measured outputs will still have noise present and a percentage error equivalent to the noise/signal ratio will always be obtained.

A similar graph to this is shown in figure 5.13, but here at T = 2.5 secs, a fault

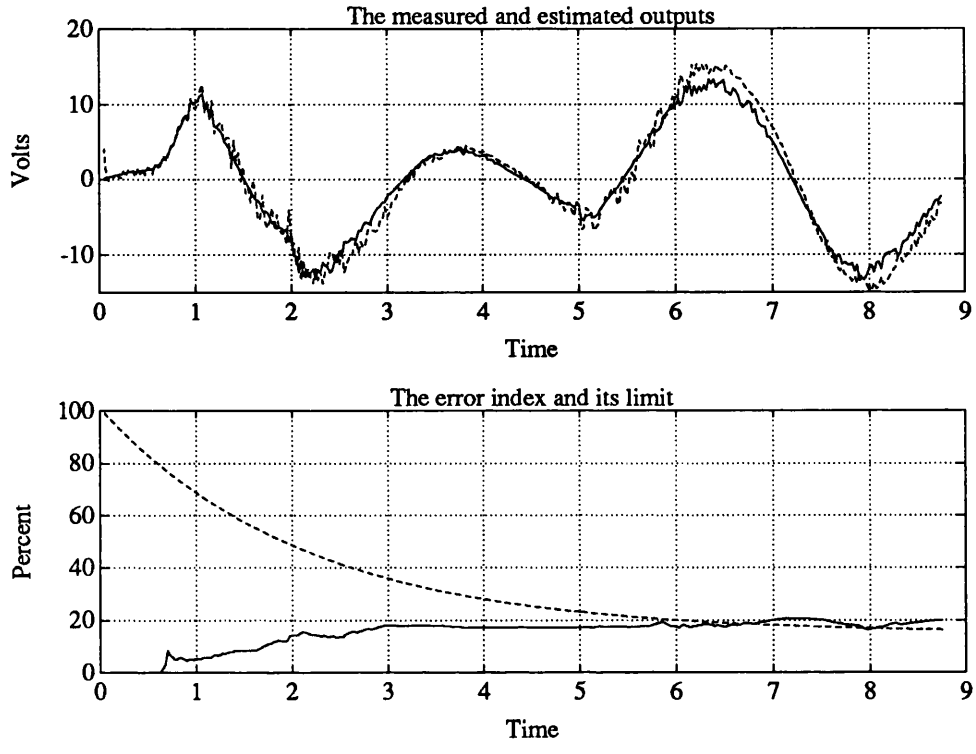


Figure 5.13: A fault appearing in the R-L-C circuit.

develops, C changes from 50mF to 67mF. The top graph shows the measured output (the full line) and the estimated output (the dashed line). The graph of the error index is shown below this. The results are that the error index can be seen to be higher than before at T=3 seconds (figure 5.12, with no fault present). Because the output level is fairly low from about T=3 to T=5, there is little change in the error index, after T = 5 the output error is approximately 20%, and at T=6.7 this becomes larger than the error limit and a fault is detected.

Figure 5.14 shows another example. Here C changes from 50mF to 333mF. A much larger discrepancy is seen in the estimated and measured output, and a larger increase in the error index is also obtained. The fault will be detected at T=5.5 seconds.

In both of these examples the detection of the fault was delayed for two reasons. Firstly, soon after the fault occurs the measurements from the system are quite small,

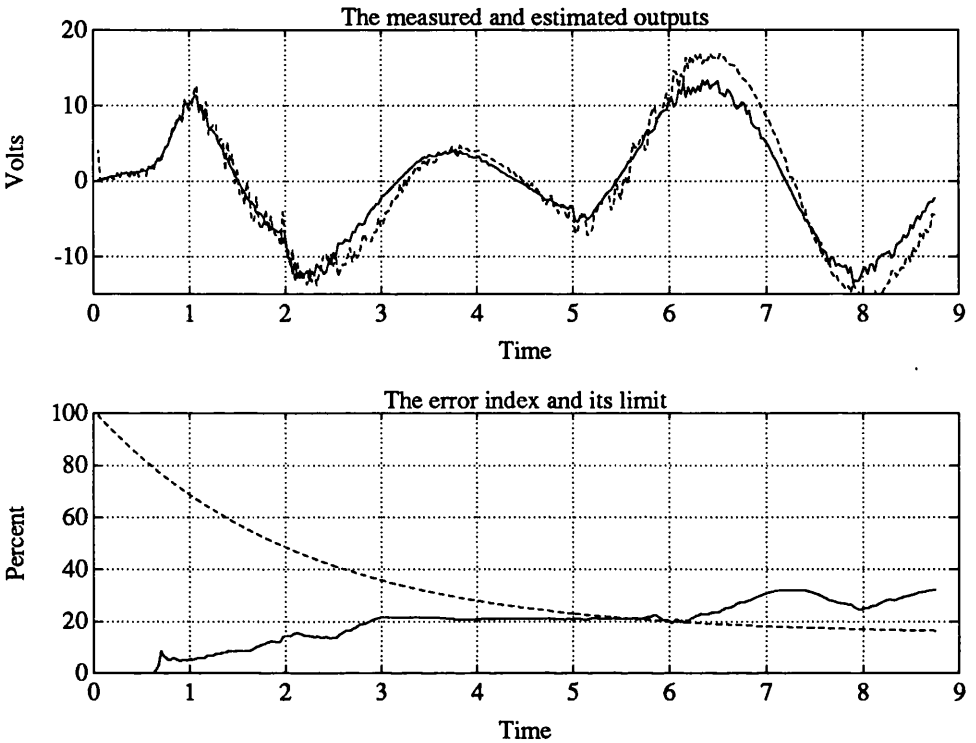


Figure 5.14: A fault appearing in the R-L-C circuit.

the noise content of these measurements is therefore higher (in proportion to the signal level), and so less attention is paid to errors as they are assumed to be from excessive noise. Secondly, when the fault occurs the assumption that the state estimates are not very good is still being made, and so an error in the outputs can be expected to be caused by this. As time progresses the certainty about the state estimates increases and the output error can no longer be explained by errors in these estimates. There must therefore be a fault in the system as modelling errors (see earlier) are inhibited from causing a fault to be detected, large measurement noise in the signals is inhibited from causing a fault to be detected and the state estimates should now be good.

If the system had been monitored for some time before the fault occurred, then the error index limit would already be at a low level and a fault would be detected much quicker. Obviously the key to this is deciding what the limit is for 'small' measurements. This should be round about (2)× (the typical value for the

measurement) × (the expected noise/signal ratio). Any measurements below this can be expected to contain a relatively large amount of noise.

An area which has not been specifically covered is that of components which have slowly changing parameters. With the methods described above, these would at first not be detected, as the slight change in the system behaviour which they would cause could be accounted for by modelling errors and measurement noise. However, as the parameter value moves further from its original value, the change in the system's behaviour will increase and the error index measurement will gradually increase. When the error index exceeds its limit a fault will be indicated.

Assuming that it is possible to identify this component and find a new value for its parameter, then this new value can be used to modify a controller for example and it can also be inserted into the model of the system, so that the model is correct again. Monitoring the system can then restart. If the parameter is still slowly moving, then at some point in the future it will cause another fault to be detected. Of course, the operator should be informed about the situation and he should make the decision about what action to take.

Finally, it was stated at the beginning of this chapter that the measurements used had not first been passed through the prefilter described at the end of chapter 4. This was to demonstrate the abilities of the filtering described in this chapter without the influences of any other filters. For completeness the results discussed earlier with those obtained when prefiltering is used will be compared.

Figure 5.15.A shows two error indexes. The dotted one is the one seen in figure 5.12 and is the error index without prefiltering when no fault is present. The full line is the error index when prefiltering is used. It can be seen the prefiltered error index is lower than unfiltered one and it is more 'even'. This means that the possibility of a false alarm due to noise is further reduced. Figure 5.15.B shows the same two error

indexes but this time for the fault at T=2.5 secs when C changes from 50mF to 333mF discussed earlier (figure 5.14). Here the prefiltered error index remains at a lower level until the size of the output signal increases (and its proportional noise contents drops), then the error index rises sharply. The detection time is slightly increased, but this is due to the lower level of the error index before it became apparent that a fault was present. By using the prefilter the certainty about detecting a fault is increased, since the error index is less likely to increase above the error limit because of the measurement noise.

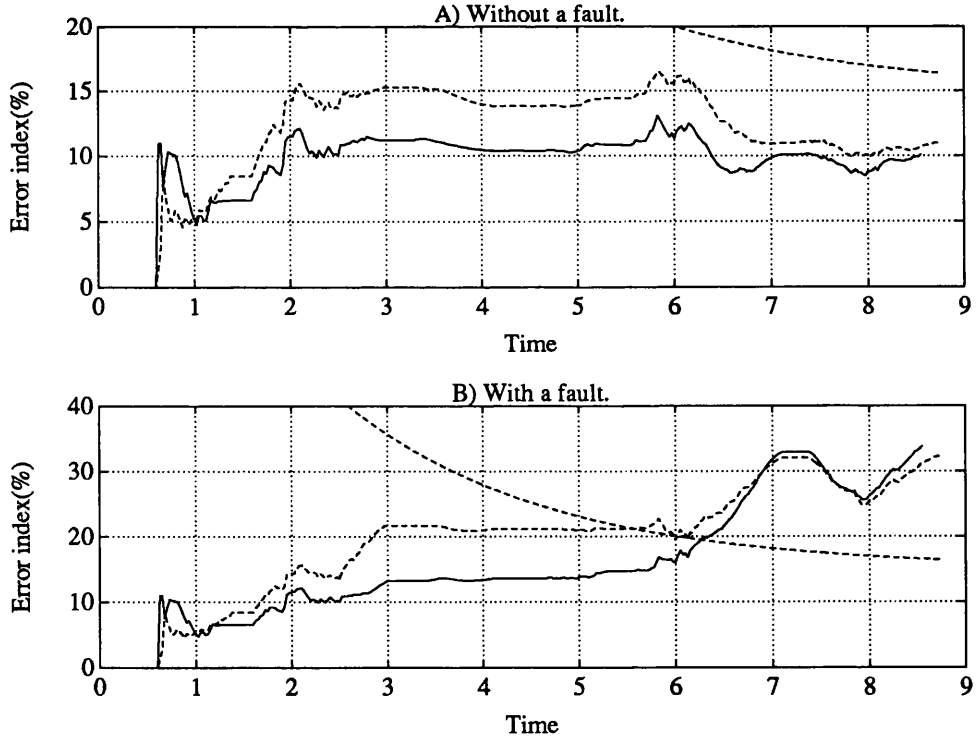


Figure 5.15: Comparing error indexes when using a prefilter on the measurements.

5.5 Summary.

Using the methods described above the occurrence of false alarms due to noisy measurements, modelling errors and poor state estimates can be prevented. This

involves calculating the states in two ways, one way based on the inputs and another way based on the outputs. These can then be combined in different proportions to find an estimate of the states. The proportion of each that is used determines the size of modelling errors that can be coped with.

Obviously, if modelling errors are large, then it will not be possible to detect small faults i.e. small changes in the parameter of one component. This is also true if noise levels are high. However, if a noisy system which is not well known develops small faults, then there is no way that detection by any means would be possible. In any case parameter changes of this size would be likely to have an insignificant effect on the behaviour of the system. If it was desired to detect small faults in these kinds of system then there would be a price to pay in that many false alarms could arise.

Chapter 6

Fault Diagnosis.

In this chapter the diagnosis of faults will be looked at, after first ascertaining that a fault is present. This will be approached from a similar angle to that of Reiter [68], except that it is not possible to implement his theory of diagnosis directly for dynamic systems. One of the main reasons for this is that his theory requires the use of a "*theorem prover*" for the particular domain in which the diagnosis is to be performed. A theorem prover is essentially an algorithm which can check a multiple fault hypothesis[1] against observations from the system and then report back saying whether or not the hypothesis was consistent with the observations.

In addition to finding the location of a fault, the size of a fault is also to be calculated. For example, not only should the diagnosis algorithm report that "*pipe B is faulty*" but also that "*its effective diameter has changed from 0.1 metres to 0.062 metres*". In this case possibly reflecting the fact that pipe B has become partially blocked.

For any particular fault which develops in a system, it is often possible to come up with more than one possible diagnosis, all of which are equally likely based on the observations from the system [8] [36]. The number of diagnosis can sometimes be

[1] A multiple fault hypothesis consists of 1 or more components which are considered to be faulty.

reduced by making further observations, or it may be necessary to have more sensors on a system in order to differentiate between different competing diagnoses [11] [10] [68]. If it is not possible to obtain further information from either more observations or more sensors, then a number of valid diagnoses must be checked manually on the system itself. Even in this case, fault diagnosis can still be of great use. Narrowing down all of the possible faults on a complex plant to just a handful which need to be checked manually could be of great help in getting the plant operational again.

A further point to note is that since noisy systems are being dealt with, and the model of the system is not 100% accurate, it is never possible to say with complete certainty that a hypothesis matches the observations or that it definitely doesn't match them. A generalised likelihood approach will therefore be adopted for determining how well a hypothesis matches the observations. A value of close to 1 indicates a high likelihood that a hypothesis does match the observations and a value of about zero indicates a low likelihood of a correct hypothesis.

We will begin by looking at how to create a "*theorem prover*", as defined by Reiter, for the domain of dynamic systems, initially without the addition of noise or modelling errors. It is noted that the assumption that *components which are behaving abnormally are consistent in their abnormality*[2] is useful, and doing this enables one to greatly increase the depth to which a faulty system can be investigated and increases the number of simultaneous faults which can be considered.

Following this it will be shown how noisy measurements affect this and the implications of modelling errors will be discussed. It is found that Reiter's theory of diagnosis is not practicable, because of the real time, on-line, recursive nature of the fault diagnosis undertaken here. However, some of his work is utilised: subsets of multiple fault hypotheses are examined as are "conflict sets" to reason about whether other hypotheses can be regarded as correct.

[2] i.e. the nature of the fault is constant and does not fluctuate

How to find the most likely diagnoses will then be looked at, and some of the problems with implementing such a system, in real time will be examined. Some solutions to these problems will be proposed, most significant of which is a method of reducing the number of conflict sets to be tested, whilst maintaining the ability to diagnose the same number of faults. This significantly reduces the computing power needed, whilst not greatly degrading the diagnosis process for any real system.

6.1 Theorem Prover for Dynamic Systems.

A *theorem prover* as described by Reiter [68], must be able take a system description, observations of that system, and a subset of those system components which are assumed to be behaving correctly. The *theorem prover* must then decide whether the observations of the system are consistent with the set of components behaving correctly. If the set of components is not consistent then this set is termed a 'conflict set' [9], that is, this set of components when considered to be behaving correctly is in conflict with the observations of the system.

Reiter summarised this as :-

$$SD \bigcup OBS \bigcup \{\neg AB(c_1), ..., \neg AB(c_k)\} \tag{6.1}$$

is inconsistent.

Where SD is the system, OBS are observations of the system, $\{c_1, ..., c_k\} \subseteq COMPONENTS$ and $COMPONENTS$ are the components of the system. $\neg AB(c_n)$ means component c_n is not abnormal.

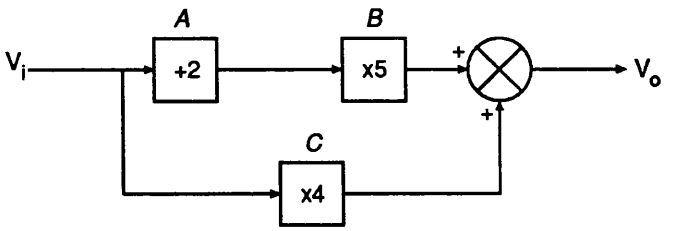
A correct diagnosis is then a minimal hitting set for all of the conflict sets. A hitting set, H, on a *set* of conflict sets, F, is a set whose intersection with each member of F (i.e. with each conflict set) is not the empty set. A minimal hitting set on F has no proper subset which is also a hitting set on F.

A point which should be noted is that a conflict set is determined by considering a subset of components which are assumed to be behaving correctly, and which are in conflict with the observations. This does not make any assumptions about the components of the system which are not in this subset. i.e some of the components not in the subset may be behaving normally or abnormally, but this is not of concern at the moment.

This theory of diagnosis was derived with reference to ideal non-dynamic systems. Although it is valid for all domains of systems, there are a number of aspects of this theory which, when applied to dynamic systems, are not as straight forward as it would first seem. There are also a number of practical issues which need to be considered.

Equation 6.1 implies that it is only necessary to consider a subset of components, which is true if there are many sensors on the system being monitored. However, if the number of sensors is limited, as is true in many cases, then only considering a subset of components and ignoring the rest of them will lead to the situation where there is very little that can be determined about the system. For example in figure 6.1.a a simple system and two observations of the system are shown. Both of which are in conflict with the system's description. If equation 6.1 is used directly as implied, then when the subset of components considered is $\{A, B, C\}$, it can be seen to be a conflict set. If set $\{A, B\}$ is considered, then the result is that it is not a conflict set. i.e. if A and B are considered to be functioning correctly, and the behaviour of C is irrelevant, then the observations can be considered as being consistent (figure 6.1.b and 6.1.c). This can be repeated with the sets $\{A, C\}, \{B, C\}, \{A\}, \{B\}$ and $\{C\}$, and then there is only one conflict set $\{A, B, C\}$. This set has three minimal hitting sets $\{A\}, \{B\}$ and $\{C\}$, and there are therefore three correct diagnoses of this system for these observations. So all that is known is that a fault in any one of the

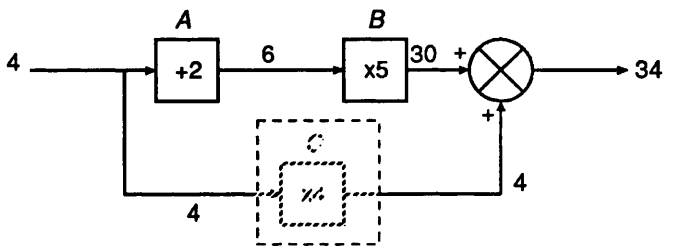
components A, B or C would explain the observed fault.



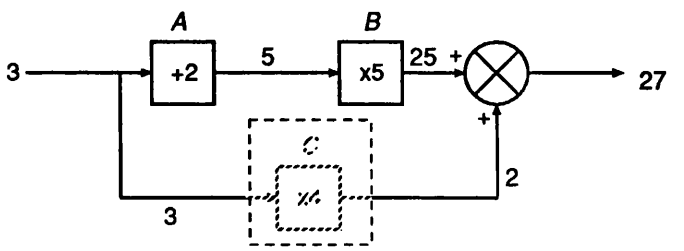
Observation 1: $V_i = 4$, $V_o = 34$.

Observation 2: $V_i = 3$, $V_o = 27$.

(a). The system and two observations.



(b). Testing {A,B} for conflict with Observation 1.



(c). Testing {A,B} for conflict with Observation 2.

Figure 6.1: A simple system consisting of components with parameters.

6.1.1 Diagnosing Faults in Parameterised Components.

Doing this did not take advantage of all that could be deduced about the system. When Reiter[68], de Kleer and Williams [10] [11] were working towards this theory of diagnosis, they did so using parameter-less components, they worked with binary gates and systems of adders and multipliers, i.e. each component had a function

but no parameter. Here real, dynamic systems are being dealt with, and so some of their properties can be taken advantage of. Firstly, the actual way a real component operates is based on physical laws which do not change, so even if a component is behaving abnormally, it is unlikely to be behaving in a completely unrelated manner to it's original behaviour. A fault in a component will therefore be assumed to be comparable with just a change in the value of its parameters. To further back this up, it is noted that if the behaviour of the component had fundamentally changed, then it is very difficult to try to find out its new fundamental behaviour. To summarise, the only diagnosis attempted is in components, when the fault manifests itself as a change in the parameter of a component, and the new value of the components parameter can be regarded as being constant during the diagnosis process.

Reiters theory of diagnosis did not specify the need for evaluating a faulty component's new behaviour. It will be seen later however that this is not only desirable from the point of view of knowing more about a fault, but it is also essential if the system is noisy and the model of the system is inaccurate. It is necessary to find out how faulty components are behaving (and therefore the new behaviour of the system as whole) in order to be able to say if a diagnosis is likely to be correct (i.e. to locate the fault). Assuming this new behaviour is constant, further observations from the system can then be used to see how well they match the diagnosis.

In order to locate a fault, the new behaviour of a suspect component must be evaluated. This in turn means that the new value for its' parameter must be evaluated. Now re-examine figure 6.1.a. Wherever a set of components would normally be tested, it is instead assumed that the components in the set are behaving correctly, and the other components have unknown parameters which must now be identified. In table 6.1 the result for these two observations are shown. For the set $\{A, B, C\}$, it is assumed that all of the components are working correctly, and therefore there are no

component's parameters which need to be identified[3]. For the set $\{A, B\}$ it is assumed that A and B are working correctly and a parameter for C is then evaluated to make the model of the system consistent with the observations. Firstly using observation 1, the value for component C's parameter must be 1.0 as indicated in the table. Secondly, for observation 2, the value of C's parameter is 0.667. This is repeated for in a similar fashion for $\{A, C\}$ and $\{B, C\}$. Remember, using Reiters method directly revealed that there was a fault in either A or B or C, but looking at table 6.1 it is seen that when the parameters of components have been evaluated, there is only one which remains the same for both observations. That is when considering component $\{B\}$ to be faulty, with a value of 3 for it's parameter.

	Observation 1 Parameters			**Observation 2 Parameters**		
Subset	**A**	**B**	**C**	**A**	**B**	**C**
$\{A, B, C\}$	-	-	-	-	-	-
$\{A, B\}$	-	-	1.0	-	-	0.66
$\{A, C\}$	-	3.0	-	-	3.0	-
$\{B, C\}$	-0.4	-	-	0.0	-	-
$\{A\}$	-	$5.667 - 0.667C$	C	-	$5.4 - 0.6C$	C
$\{B\}$	$2.8 - 0.8C$	-	C	$2.4 - 0.6C$	-	C
$\{C\}$	$\frac{18}{B} - 4$	B	-	$\frac{15}{B} - 3$	B	-

Table 6.1: Two observations of a system which has parameters.

For the sets $\{A\}$, $\{B\}$ and $\{C\}$ in table 6.1, it is not possible to evaluate the parameters of two components using just one observations, as this only gives enough information to find a relationship between the two suspected faulty component's parameters. These correspond to multiple faults. Using two observations however makes it possible to solve these relationships and find the value of the parameters which will make the model consistent for two observations. In order to be able to compare these a third observation is obviously needed, and then the parameters must

[3]Since a fault will have already been detected by this stage, this set would not actually be tested since it is known that it must be inconsistent for any fault to present.

be re-evaluated using the second and third observation. If a third observation was $V_i = 1$ and $V_o = 13$, then the results would be as shown in table 6.2. This is analogous to propagating signals through time, as discussed in chapter 4.

	Observations 1 & 2 Parameters			Observations 2 & 3 Parameters		
Subset	**A**	**B**	**C**	**A**	**B**	**C**
$\{A\}$	-	3	4	-	3	4
$\{B\}$	1.2	-	2	1.2	-	2
$\{C\}$	2	3	-	2	3	-

Table 6.2: Using three observations to solve the parameter values.

$$V_o = V_i(B + C) + (A.B) \tag{6.2}$$

6.1.2 Practical Limitations on Hypotheses and Conflict Sets.

In the table above the value of the parameters remain the same. This is because the behaviour of the system can be described by equation 6.2, so as long as $(B + C) = 7$ and $A \times B = 6$ then if one parameter is defined, all the others are known. The same situation arises in dynamic systems, where the behaviour of the system is described by a transfer function, where the ratio between the parameters of the transfer function are important, but the actual value of the parameters are not. For example, with a DC motor where just the voltage and angular velocity are measured, the system has the following transfer function which describes the dynamics of the system.

$$\omega = \frac{K}{(JL)s^2 + (JR + LC)s + RC + K^2}.V_{in} \tag{6.3}$$

From the systems dynamics it is only possible to identify the lumped parameters $\frac{JL}{K}$, $\frac{JR+LC}{K}$ and $\frac{RC+K^2}{K}$. So there are three equations in the five parameters of the

system.

For a fault hypothesis which contains one component e.g. either J,L,R,C or K, it is possible to find whether or not it describes the fault and if so, what the value of its parameter must be.

For a hypothesis which contains two components, e.g. (R,L) or (J,K) it is still possible to find out whether the hypothesis can describe the fault but with the difference that there may be two possible answers for the hypothesis because the solution may be quadratic. e.g. if (R,L) were faulty, one and only one value for R and L can be found which would describe the fault, but if (J,K) was the fault then a set of two solutions for the parameters J and K can be found, each of which would correctly describe the new dynamic behaviour of the system. So, although the fault may have been located, it cannot be uniquely identified without further information.

For a hypothesis which contains three components e.g. (R,L,J) or (R,K,C), there is a further problem. The problem that some hypotheses will have multiple solutions still exists, but also a new problem arises in that every combination of three components will explain the fault correctly. For example if there was a fault in R, L and J in the dc motor, new values for R, L and J could be found which would explain the fault correctly, however new values for R, K and C or any other combination of three components could also be found which would also explain the fault correctly. This is because the dynamics of the system are described by the three parameters in the transfer function but an attempt is being made to find three of the five physical parameters assuming the other two of the physical parameters are unchanged. Any three of the physical parameters can therefore be taken and values found for them which do correctly describe the systems dynamics, and there is no way of telling if these are actually the faulty components or not. There is then no point in testing a fault hypothesis containing three faults for this type of dynamic system since the

result will always be true. This is also true for hypotheses containing four or five components.

If access to a larger number of measurements is possible, for example the supply current in the dc motor example, then it is possible to find more about what is going on in the system. More measurements from the system means that the problem can be broken down into smaller sections and more information can be gained about what is happening in the system as a whole. An example of system identification on a DC motor can be found in [60], here measurements of the input voltage, the supply current and the rotational velocity of the motor are used to identify the system's parameters. If, however, only the voltage and rotational velocity were available, then identifying all of the systems parameters is not possible.

So far only an outline of how it is possible to calculate the values of the parameters of those components not listed in a particular conflict set has been given, i.e. the components assumed to be behaving incorrectly. As already stated, if the parameters remain constant throughout each observation of the system, this indicates that a fault in those components not contained in the conflict set will explain the observations from the system. Alternatively, a set of components is only a conflict set when the parameters of the components excluded from it cannot be evaluated to be the same value for each observation.

Looking back to table 6.1, it is seen that constant parameters for the components excluded from $\{A, B, C\}$, $\{A, B\}$ and $\{B, C\}$ could not be calculated i.e. when it is assumed that the components in each one of these sets are not abnormal, then this means it should be possible to calculate the parameters of the remaining components and they should be constant throughout each observation. Since this cannot be done, the components in each of the three above sets cannot all be behaving correctly, and each of these sets must therefore be a conflict set. For all of the other sets in table 6.1

it was possible to calculate constant parameters for the components not included in the sets, so they are not conflict sets. The minimal hitting set for $\{A, B, C\}$, $\{A, B\}$ and $\{B, C\}$ is just $\{B\}$, and so there is only one diagnosis for the fault and that is that B is faulty. The new value for the parameters for a diagnosis are obtained from that conflict set which had the components in the diagnosis excluded. In this case it would be the set $\{A, C\}$ from table 6.1, since this set does not contain B, but does contain all the other components of the system. The new value for B's parameters is 3.0.

As a suspected conflict set becomes smaller, then the number of components which are suspect increases, and the procedure for finding values for their parameters becomes more and more complex. If it is not possible to calculate the parameters then the assumption must be made that the set of components is not a conflict set, since it was not possible to prove that it is. This therefore puts a limit on the number of components that can be diagnosed to be simultaneously failing. In the DC motor mentioned earlier, only a maximum of two components failing simultaneously could be diagnosed, since when 3 components are faulty, the values of any three component parameters can be chosen to fit any new system behaviour, and if 4 or 5 components are faulty, then the equations cannot be solved.

This has shown the basic principle for a *Hypothesis Evaluator* for dynamic systems, or indeed any system which is comprised of components which contain parameters. Noise was not considered nor modelling errors. These two factors force a change in the way in which a set of components is determined to be a conflict set or not. This is simply because when noise is present, the calculated value for parameters will never be exactly correct. Using the test outlined above, (i.e. that the calculated parameters have different values for each observation), will always be true and every set of components will always be a conflict set. This is the topic of the next section.

6.2 Finding Conflict Sets for Dynamic Systems.

In the previous section an outline was given of one method for determining conflict sets for dynamic systems, without the presence of modelling errors or noise in the measurements from the system. How these affect the theorem prover from the previous section, and how they must be adapted will be investigated. It will also become clear why the new behaviours of faulty components need to be evaluated.

As stated before, when noise or modelling errors are present, the value of a parameter needed to make one or more observations consistent will never be exactly the same for each observation, therefore a test for equality is irrelevant. As an alternative, a test for how approximately equal the parameters are could be used. When considering parameters for different components which could have values which are completely different orders of magnitudes, then making comparisons between how approximately constant they are becomes difficult. Instead of doing this, a method will be used which will recursively in time try and make an estimate for one or more component's parameters, given that the other components are behaving correctly. e.g. to test if $\{A, B\}$ was a conflict set, then the value of C's parameter needs to be recursively estimated. If it were required to test if $\{A\}$ was a conflict set, then the value of both B and C's parameter would need to be recursively estimated.

Once an estimate has been made of any unknown parameters, they can be used in a model of the system which uses the known parameters for the components which are assumed to be correct, the estimated parameters for the components which are assumed to be faulty, and the system's inputs as measured. Using this information it is possible to predict what the system's output should be given this information[4]. If the predicted output is not significantly different from the actual output, then the components which were assumed to be faulty do correspond with the systems

[4]This is a slightly simplified description in that for dynamic systems an estimate of states is also needed. This is dealt with later.

new behaviour, and the component set being considered is not a conflict set. If the predicted and actual outputs are significantly different, then the component set being considered is a conflict set. We will shortly describe how to determine significant difference between the outputs, but for now, an insignificance difference returns a value of close to zero, while significant difference gives a larger value. A value of 1 or more means the outputs are completely different. The overall outline for all of this is shown in figure 6.2.

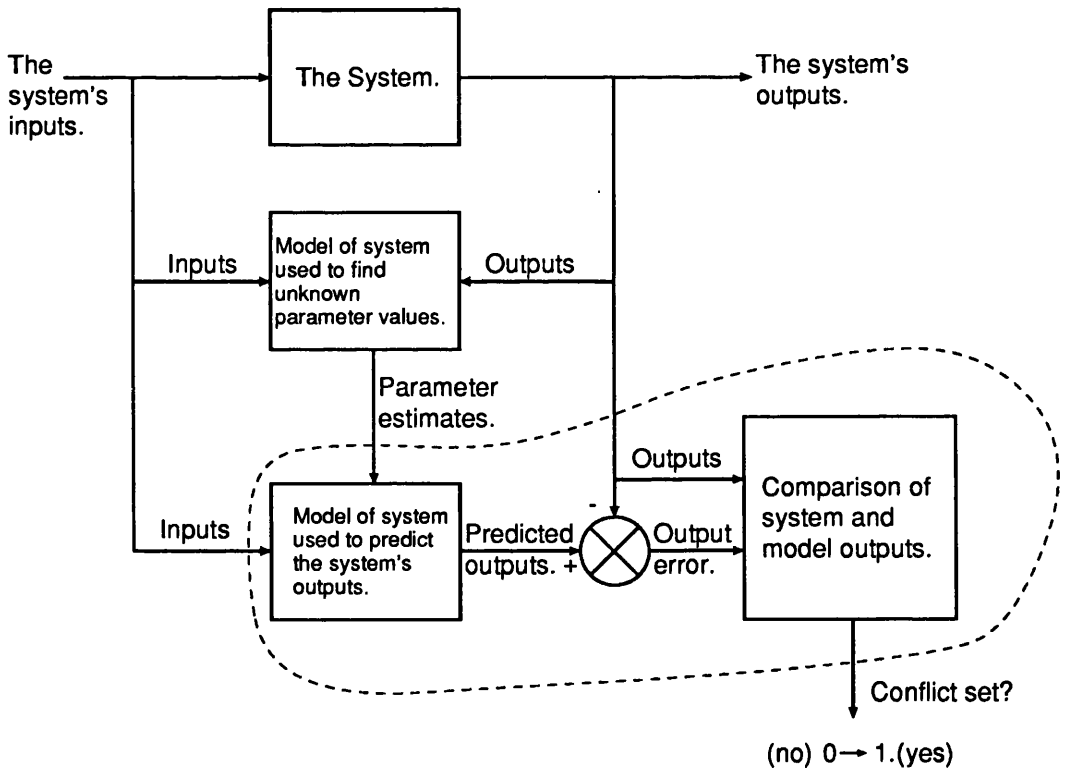


Figure 6.2: The basic arrangement of the Hypothesis Evaluator.

An example using a dc motor will now be discussed. Measurements of the applied voltage V and the angular velocity of the motor ω will be used. Two sets of components will be examined. One will be a conflict set, and one will not be. The objective will be to determine which is which. This will firstly be done with no noise and no modelling errors, then the case when noise is present is looked at, and finally the case when both noise and modelling errors are present is shown.

6.2.1 Finding Conflict Sets in Noise Free Systems.

Figure 6.3 show the applied voltage (the dashed line) and the measured angular velocity (the full line), samples are taken every $\frac{1}{50}^{th}$ of a second.

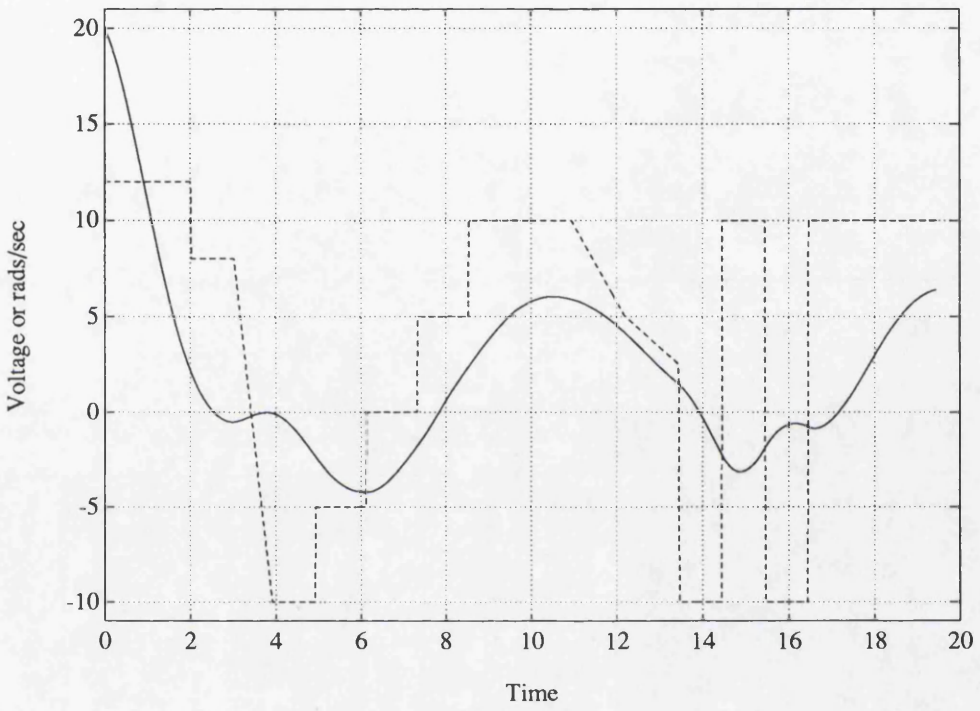


Figure 6.3: A simulation of a DC motor.

The two sets which will be looked at are $\{R, K, J, C\}$ and $\{L, K, J, C\}$. Here, it is assumed that the presence of a fault has just be detected, and diagnosis is about to start. The model of the system is the same as that discussed in chapter 4. The values of the parameters in the model of the system are shown in table 6.3.

Component	Parameter
R	3
C	0.4
L	2.5
K	2
J	1.5

Table 6.3: The component's parameter values.

Firstly the case when there is no noise is examined. When the component set

$\{L, K, J, C\}$ is considered, it is assumed that these four components are behaving correctly and that component R is faulty. The first step is then to find a value for R's parameter. Figure 6.4 shows the value that component R's parameter must adopt in order for the observations to be consistent. At the moment R is not recursively estimated, a value for R for is merely calculated for each time interval. This is based on the current observation and 2 earlier ones using the propagating signals through time method as discussed in chapter 4. It is seen, from looking at the graph of R, that the calculated value for it remain essentially constant. This means that the assumption that $\{L, K, J, C\}$ could be functioning correctly was correct and that R is faulty, since the value of R's parameters has changed and is constant. $\{L, K, J, C\}$ is therefore not a conflict set.

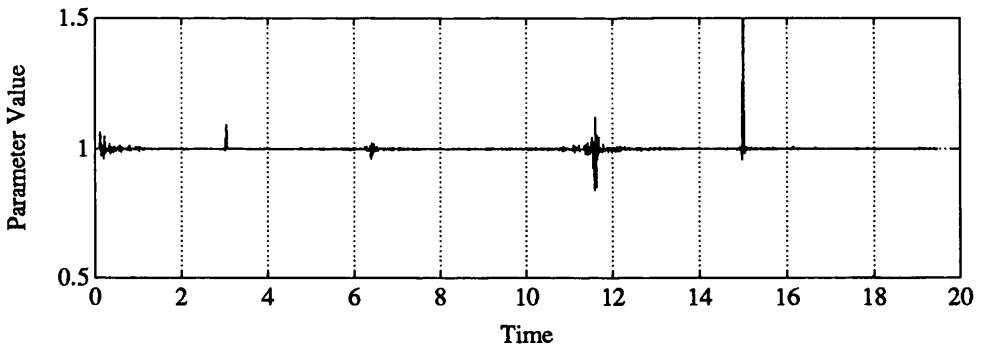


Figure 6.4: The values of R needed to make the observations consistent.

Figure 6.5 shows the result when looking at the set $\{R, K, J, C\}$, and the value that component L's parameter must adopt in order for the observations to be consistent is calculated. Clearly L's parameter is not constant, it must change value as the measured inputs and outputs change in order to keep the observations consistent. This means that the assumption that $\{R, K, J, C\}$ are all functioning correctly was incorrect and that $\{R, K, J, C\}$ is therefore a conflict set.

This was just an example to demonstrate how to identify a conflict set, in its simplest form. When dealing with noisy measurements it is necessary to do a lot

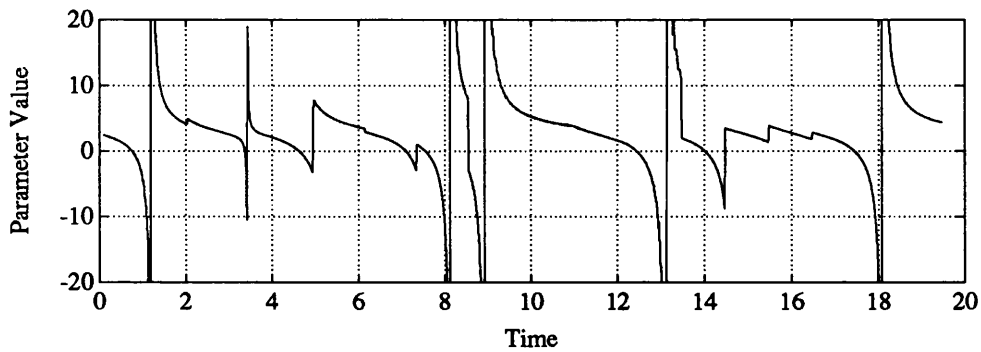


Figure 6.5: The values of L needed to make the observations consistent.

more work. Simply using the propagation through time model results in very noisy parameter values needed to make the observations consistent. To get around this an attempt is made to reduce the noise by firstly pre-filtering the measured signals, as described in section 4.6. The states of the system are then estimated, and these are used to help estimate the values of the parameters. Having obtained some estimates of parameter values, these will be used to help improve the state estimates. The value for a components parameters will no longer be just the value needed to make the current observation consistent, but it will be a least-squares estimate based on all of the observations so far obtained. This has two implications.

6.2.2 Finding Conflict Sets in Noisy Systems.

Firstly, when diagnosis begins, the parameter estimates for any of the components are likely to be poor, and they will be noisy. At the start of the diagnosis all of the parameter estimates for each suspected conflict set will not be constant, and therefore determining which are conflict sets will be difficult.

Secondly, as time continues, the parameter estimates should get better as they are in some sense an "average" of all the previous estimates. Because the estimates are now averages of previous observations, they are no longer able to quickly change their value when the observations change. So the type of result seen in figure 6.5 will no

longer be possible, instead the result would be an average of this. This means that as time progresses all of parameter estimates become essentially constant so using the method as described for the no-noise system would yield the result that there were no conflict sets and therefore no fault!

Instead of checking to see if parameter estimates are constant, the method outlined in figure 6.2 will be used. The parameter estimates will be used to predict what the value for the system's outputs should be and if the predictions are accurate then the set of components under examination is not a conflict set, however, if the predictions for the system's outputs are poor, then it is likely that the set under consideration is a conflict set. To do this, estimates of the states in the prediction models must also be kept.

This is essentially the same algorithm as used in chapter 5 for detecting a fault. That algorithm essentially determined how accurately a model of the system matched the measured behaviour of actual system, which is essentially being done in the area of figure 6.2 contained in the dashed area. This dashed region can be compared with figure 5.1 which shows the basic method for detecting faults. This dashed region is the same as figure 5.1 except that parameter estimates for some of the components are used instead of the original parameter values.

The dc motor example is now looked at again, but now with approximately a 2% noise to signal ratio. The noisy measurements of the applied voltage and the velocity is shown in figure 6.6. As before an attempt is made to find an estimate for the value of the parameters for the components suspected to be faulty in each hypothesis, and only the possible conflict sets $\{L, K, J, C\}$ and $\{R, K, J, C\}$ are considered. Figure 6.7 shows the estimate for R when considering the possible conflict set $\{L, K, J, C\}$ and Figure 6.8 shows the estimate for L when considering the possible conflict set $\{R, K, J, C\}$. Both of these start off fluctuating and gradually become smoother

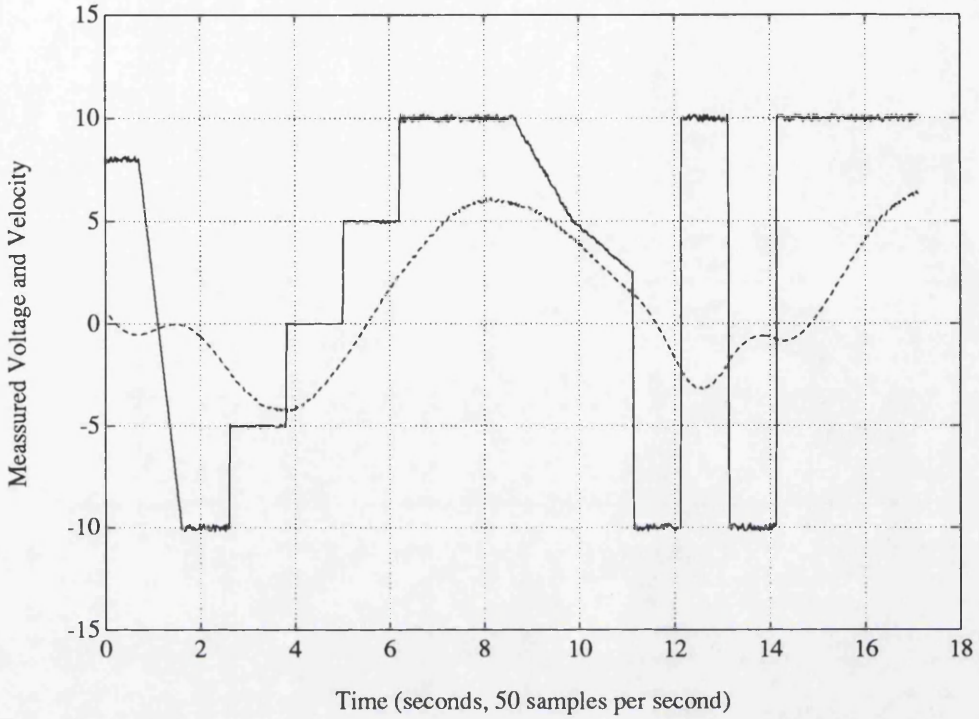


Figure 6.6: Simulated measurements of V and ω with 2% noise.

and settle out towards one value. From these alone, it is not possible to determine which component could be faulty and which isn't. (Compare these two graphs with figures 6.4 and 6.5.) These parameter estimates are those indicated on figure 6.2.

The next step in determining which subset of components is a conflict set is to take the parameter estimates for each hypothesis together with the measured inputs to the system, and produce an estimate of the systems outputs. This is part of the algorithm is the dashed region on figure 6.2.

The output estimate for $\{L, K, J, C\}$ is shown in figure 6.9. The dashed line is the actual output, and the full line is the estimate of the output using the value for R obtained at the same time period, shown in figure 6.7. The equivalent output estimate for the possible conflict set $\{R, K, J, C\}$ is shown in figure 6.10.

Finally, for each hypothesis, the estimated output is compared with the actual output. An error index is then produced in the same manner as for detecting a fault.

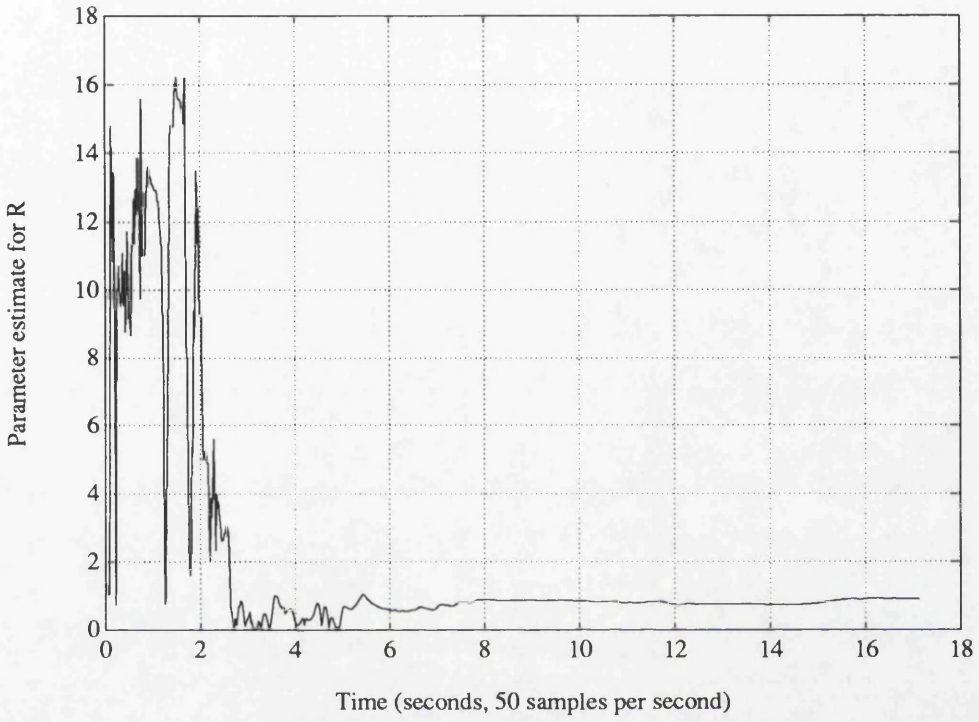


Figure 6.7: The estimate for the parameter of R.

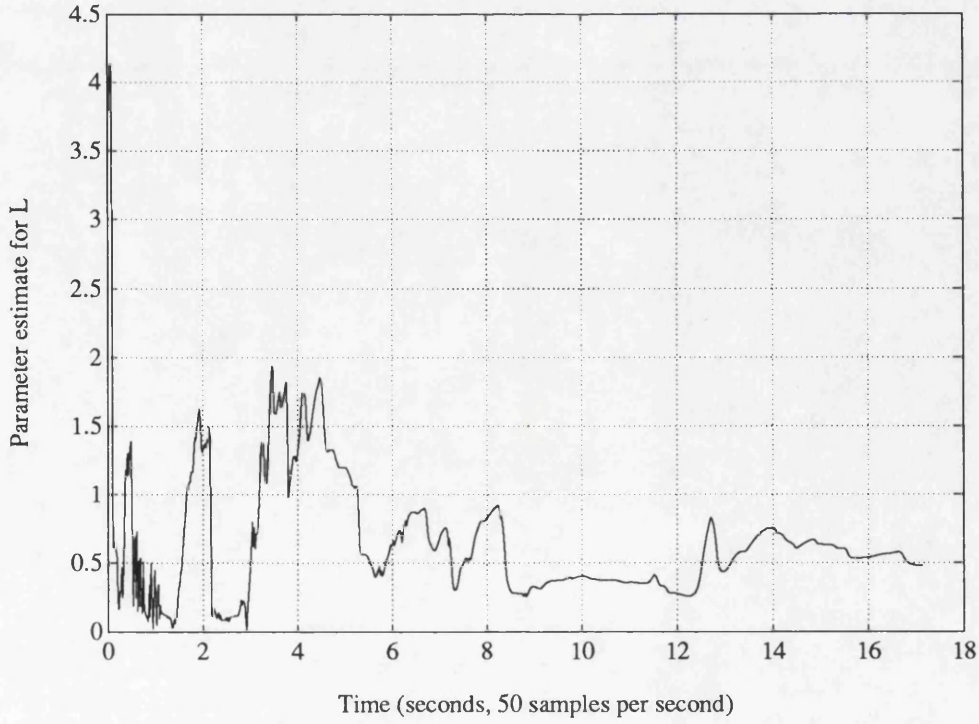


Figure 6.8: The estimate for the parameter of L.

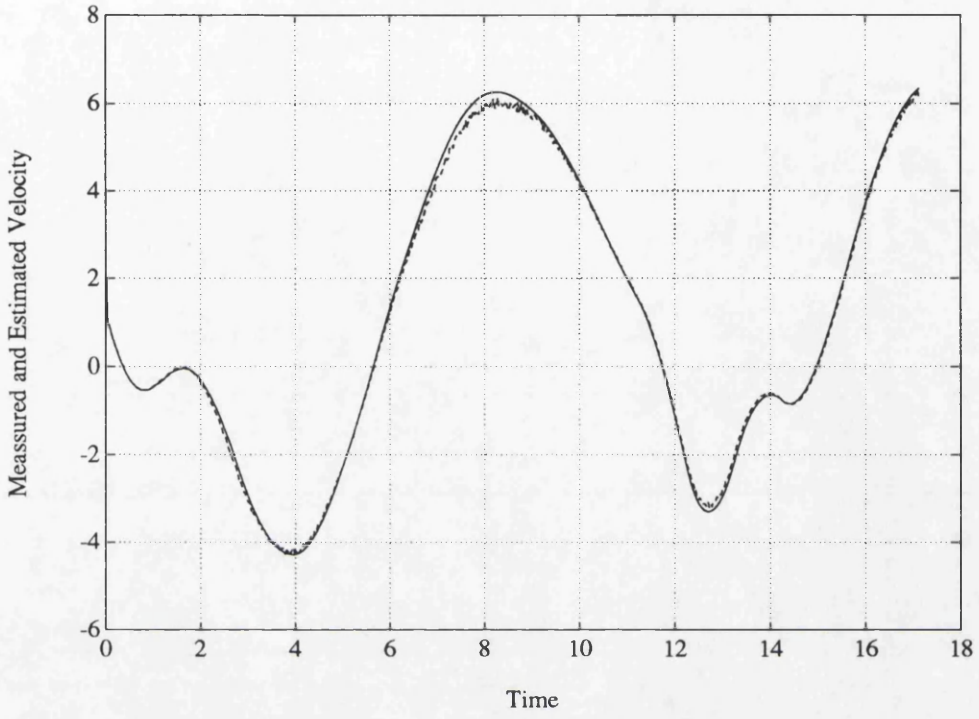


Figure 6.9: Comparing the estimated & actual output for $\{L, K, J, C\}$.

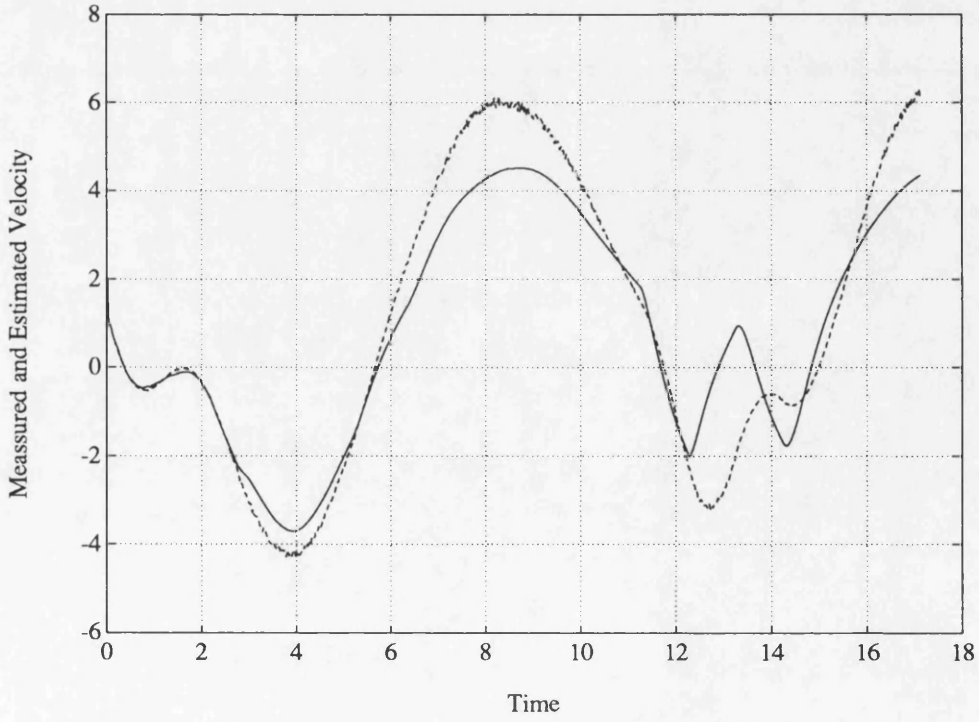


Figure 6.10: Comparing the estimated & actual output for $\{R, K, J, C\}$.

A value of zero for the error index indicates a close match with actual output, and a value of 1 or greater indicates a complete mismatch. If there is a 2% noise to signal ratio, then for the best fit possible the error index would be expected to be approximately 0.02. Bearing in mind that the parameter estimates will not be 100% accurate, when a subset of components is tested that is not a conflict set, the error index would be expected to be at least 0.02, but probably a little higher.

The error index for the set $\{L, K, J, C\}$ is shown in figure 6.11, (This is when R is considered to be faulty). After the parameter has settled down (after around 2 seconds), the error index fluctuates at between 0.02 and 0.08. The error index for the set $\{R, K, J, C\}$, (a fault is L), is shown in figure 6.12. At around $t = 2$ seconds, the error index is small, but shortly after this it rises 0.15, and then 0.4. The error index then fluctuates between 0.2 and 0.75, indicating at best a poor fit (0.2) and at worst an exceedingly bad fit (0.75), remember an index value 1.0 indicates a complete mismatch.

Using this method a measure may be produced of how "likely" it is that a set of components may be a conflict set. A high value for the error index indicates that assuming this set of components to be correct conflicts with the observations and therefore the set is probably a conflict set. An error index which is not very much greater than the noise/signal ratio indicates that the set of components is not in conflict with the observations.

In the case where there are modelling errors as well as noisy measurements, both figures 6.7 and 6.8, the parameter estimates, will look similar, although the value the parameter estimates settles to may well be slightly different. Figure 6.9,the output estimate when R is considered to be faulty, will not follow the actual output as closely, as other parameters in the system will not be accurate. The error index in figure 6.11 will therefore be similar, but it will generally have a larger value. Figure 6.10 will

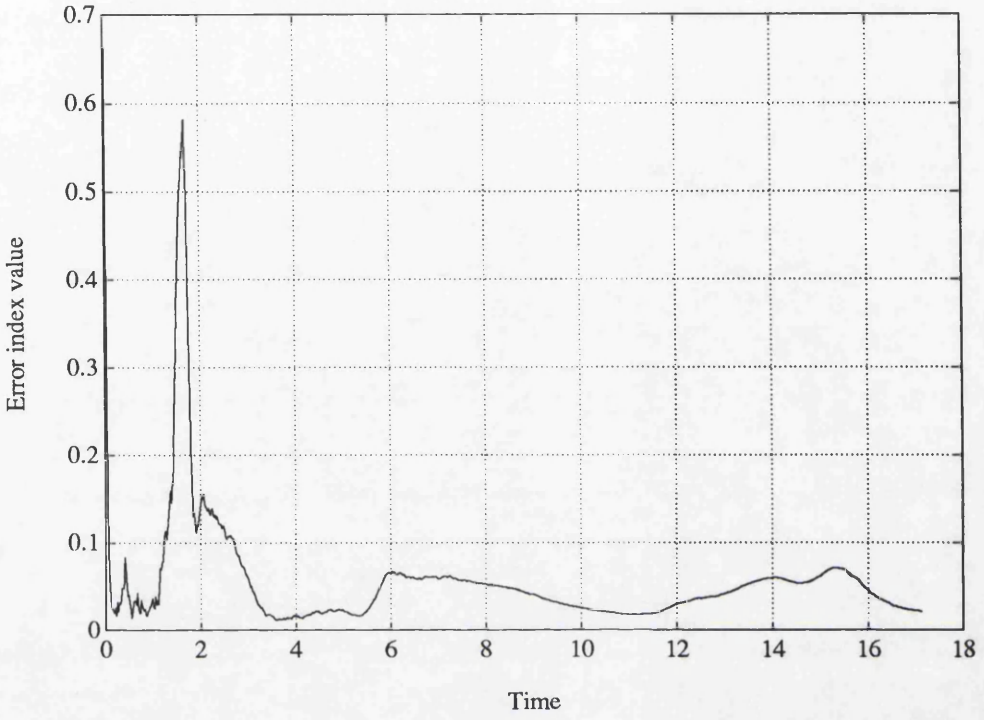


Figure 6.11: The error index for $\{L, K, J, C\}$.

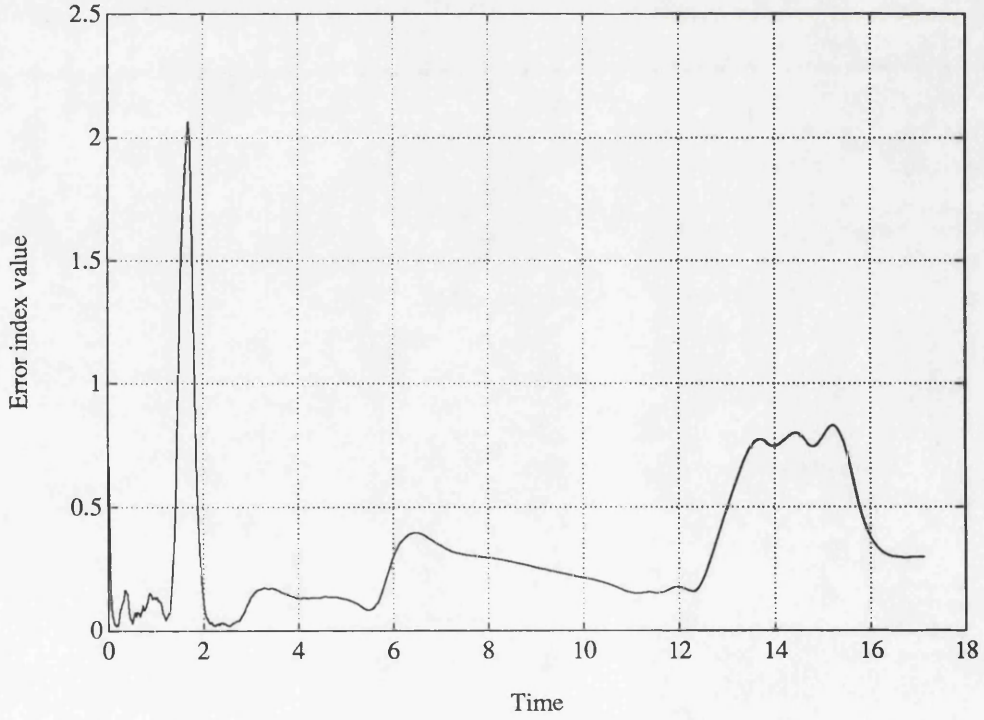


Figure 6.12: The error index for $\{R, K, J, C\}$.

still not closely follow the actual output; whether the error index is larger or smaller than before will depend in whether the fault in R has a larger effect on the output than the modelling errors. If a substantial modelling error was made in estimating Ls' parameter, and the fault which has developed in R is small, then a new value for Ls' parameter may well more closely fit the current dynamics of the system, than using the original value for Ls' parameter and a new value of Rs'. In this case it may be that the diagnosis process returns that a fault in both L and R would explain the fault. That is if the value of both L and Rs' parameter is changed in the model of the system, the model would fit the actual system more closely.

If modelling errors are small, then the net effect will just be a small increase in the error index for the actual faulty components, and a small change in the error index for all of the others. In the R L example, the error index for R in figure 6.11 will be slightly larger, and the error index for L, figure 6.12 will be either slightly higher or lower (depending upon the effect of the modelling errors). The error index for R will still be substantially less than that for L.

For every time step it is desired to check every subset of all of the components in the system, and produce an error index for each one of these subsets, then it is necessary to, in some way, find hitting sets for the likely conflict sets. This can be done either by setting a threshold of, for example, any error index which is above $5\times$ the noise/signal ratio is a conflict set. Alternatively each hitting set can be given a value of between 0 and 1 again to indicate how well it hits those sets which are likely to be conflict sets. This could be done in a straight forward algorithmic fashion, looking at potential conflict sets and at their error indices, and it would be fast and efficient when processing a large number of subsets of components. After every time step, the error index for each potential conflict set will be more accurate, and so as time continues the analysis of these error indices will become more and more accurate.

What has been shown here is that, provided the parameter values of a set of component assumed faulty can be estimated, then it is possible to determine whether or not the other components of the system (i.e. those excluded from the set of assumed faulty components) can be considered as being a conflict set.

Reiters method for fault diagnosis assumed it was possible to decide which conflict sets to test for, depending upon the result of tests of other conflict sets. This is because he said that determining whether or not a set is a conflict set is expensive computationally, which it is. However he assumed that it was possible at any instant of time to determine whether or not a set was a conflict set. (He said conflict sets would be implicitly available, i.e. the theorem prover would try to find a conflict set which meet some given criteria when requested.) Unfortunately, with a dynamic noisy system, if the diagnosis accuracy is to be improved with time, all of the observations so far obtained must be stored, and the theorem prover must then go back looking at all of this data each time it wishes to determine if a set of components is a conflict set. Doing this may mean that less accesses are made to the theorem prover (as Reiter suggests), but the amount of processing needed for each access will become, larger and larger. This is all right if in the case of off-line fault diagnosis based on a set of observations, but to perform real time, on line diagnosis doing this is clearly not possible. Instead, an error index must therefore be recursively calculated for all of the potential conflict sets which may be tested. This leads to the problem of having many error indices to keep up-to-date when the number of components in a system is large (this will be looked at shortly), but the amount of computation needed for every time step will remain constant.

We have so far just looked at an example of where there was only a single component failing. Now, the area of multiple faults is looked at more generally, before coming back to look at multiple faults in dynamic, noisy systems.

6.3 Multiple Faults

In this section some of the problem areas of diagnosing multiple faults will be discussed, which do not occur when considering only one at a time component failures. In this discussion, diagnosing multiple faults in general will be discussed, as opposed to multiple faults in dynamic systems. This will provide the basic mechanism for handling multiple component failures which will be developed and applied directly to dynamic systems. A single fault, i.e. the failure of only one component, is just a special case of multiple faults, so when the phrase 'multiple fault' is used, it means a fault consisting of one or more components which have simultaneously developed faults.

A Multiple fault occurs when two or more components develop faults at approximately the same time. Faults may be produced by a chain reaction effect, i.e. one component fails which very quickly leads to the failure of another component and so on. Alternatively the faults may be independent and it is just coincidental that they occurred at the same time. What ever the origins of the multiple fault are, the ability to identify which components have failed is needed, and if possible, the manner in which each component is behaving after failure is also most useful. For instance a pipe starting to become blocked up, will have a new effective diameter, or if a resistor which has become damaged, will have a new value (e.g. burnt out, infinity; shorted, zero; or slight damage, resistance increased by 10 %). Remember, a fault is defined as a change in a components parameter.

In particular, the problem of how to cope with finding exactly which components have failed will be discussed when the number of components is large and therefore the number of combinations of two or more components is extremely large. A method will then be introduced which greatly reduces the number of hypothesis which have to be tested, from 2^N to just N. This is important since the testing of a hypothesis is

very computer intensive. It is then shown how the method finds, in a computationally efficient way, which of the many possible fault combinations are likely to be correct.

6.4 Limiting the Number of Hypotheses.

As discussed above, it is important to reduce the number of conflict sets that need to be tested. It will be shown how this can be done in an ideal situation and how information about different hypotheses can be deduced.

6.4.1 Definition of a Hypothesis.

A multiple fault hypothesis is now defined.

Let F be the set of all the components in the system which may develop a fault. For example in a four component system. $F = \{C1, C2, C3, C4\}$. *For a particular multiple fault hypothesis, let W be a set of all the components in the system which are assumed to be working correctly. A correct multiple fault hypothesis is then given by* $F - W$, *provided that the new behaviour of the system can be explained when all the components in W are working correctly.*

That is, if a model of the system is being used that has all of the components which are listed in W behaving correctly, and the new system behaviour cannot be explained, then $F - W$ is not a correct diagnosis as at least one of the components in W must be faulty to explain the behaviour. If W behaving correctly is consistent with the observations then $F - W$ is a correct diagnosis, although it may not be a minimal diagnosis. If $F - W$ is a minimal diagnosis, then no proper subset of $F - W$ is also a diagnosis.

To explain this more clearly, examine figure 6.13 below. This is a simple example which demonstrates the definition above. The figure represents a system where the inputs and outputs of the system have just been measured. Components A,

B and D all perform an addition of their two inputs and put the result on their output. Components C and E perform multiplication in a similar fashion. Here, $F = \{A, B, C, D, E\}$.

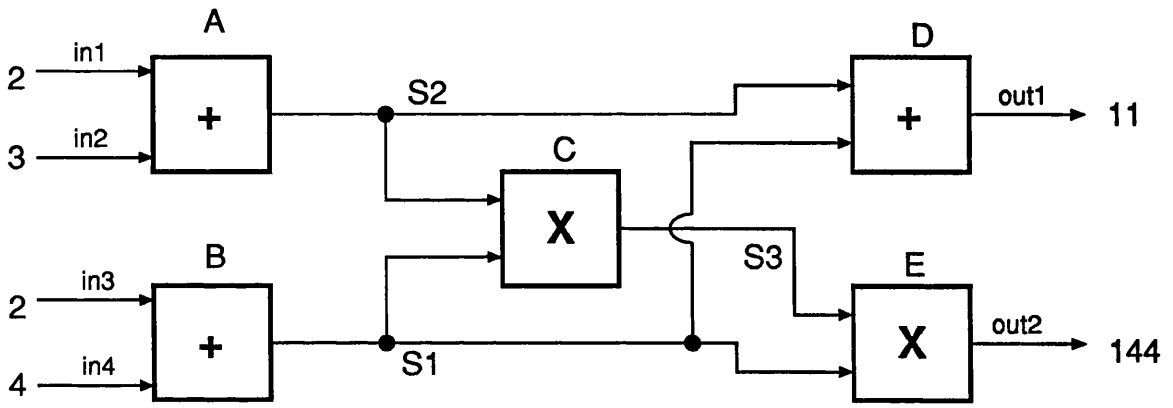


Figure 6.13: A faulty system.

As an example, suppose W, as given above, is {A, B, C, E}, that is the fault hypothesis is $F - W$, is { D }. This means that if the components A, B, C and E are working correctly, the system behaviour can be explained. This would give the following values for the signals.

in1	2	in4	4	S1	6
in2	3	out1	?	S2	5
in3	2	out2	180	S3	30

The value of the output cannot be found because it is assumed that D is faulty. The question mark means the output may have any value depending upon the manner of the fault, but looking at E the predicted output and the actual output do not match and therefore in this case the hypothesis {D} is not true, that is a fault in D and D alone does not explain the fault.

Now lets take W as {A, B, E}, which gives the fault hypothesis {C, D}, i.e. a fault in both C and D would explain the fault. The signals would be :-

in1	2	in4	4	S1	6
in2	3	out1	?	S2	5
in3	2	out2	?	S3	?

This does explain the new system behaviour in that the question marks can take any value and could therefore take the values that were measured at the output.

As a final example lets take W as {A, B, D, E}, which gives the fault hypothesis {C}, i.e. a fault in C alone. The signals in the system would be:-

in1	2	in4	4	S1	6
in2	3	out1	11	S2	5
in3	2	out2	?	S3	30

This is also a correct hypothesis.

This shows that there is a valid fault hypothesis which is {C, D} and another one which is {C}. From this it is seen that although a fault hypothesis is correct, it does not mean that every component in the hypothesis must be faulty, but rather that *if* every component in the hypothesis was faulty then the system behaviour which is observed could occur. One hypothesis which is always true is that every component in the system has failed. If this was so then any output could be obtained and hence, any outputs could be explained by this hypothesis, it is of course not a very useful hypothesis.

The important point to note here is that a multiple fault hypothesis which does explain the new behaviour of the system does not necessarily mean that every member of the hypothesis is at fault.

6.4.2 Testing the Minimum Number of Hypotheses.

We will now examine how, with a four component system just four hypotheses can be tested to find out which of fourteen hypotheses could be correct. This will also be examined in relation to conflict sets and hitting sets as described earlier.

Let the four components be named A, B, C, and D. Lets also suppose that it is possible to take any hypothesis and find out whether or not it explains the system behaviour. Lets take four hypothesis, namely:- {A, B, C} , {A, B, D} , {A, C, D} and {B, C, D}. Each one of these hypothesis assumes that three of the four components are faulty. If the hypothesis that {A, B, C} are faulty is tested, it is equivalent to testing whether the members of the set {D}, working correctly, is in conflict with the observations. i.e. the components of the system which are not included in the fault hypothesis. This set of components is termed a conflict set if the assumption that all of its members are working correctly can be shown to be in conflict with the observations. This means that the assumption is false, and therefore one or more of its' components must be faulty. Testing each hypothesis would give the results in table 6.4.

Hypothesis	Conflict Set	Explains behaviour	Conflict Set
{A, B, C}	{D}	yes	no
{A, B, D}	{C}	no	yes
{A, C, D}	{B}	no	yes
{B, C, D}	{A}	yes	no

Table 6.4: Example of a multiple fault diagnosis.

Looking at this there are two hypothesis which explain the behaviour, {A, B, C} and {B, C, D}. From this it can also be deduced that {B, C} would also explain the systems behaviour as {B, C} is a common subset of both {A, B, C} and {B, C, D}. What is being said is that ' if components B and C had become faulty, then a test of

hypothesis {A, B, C} would be true and a test of {B, C, D} would also be true'. It is conceivable, that there could be a system where both {A, B, C} and {B, C, D} could both explain a systems faulty behaviour and {B, C} if tested, would not. Bearing this in mind it is not appropriate to say that the fault <u>is</u> {B, C}, but rather that the fault is likely to be *either* {B, C}, {A, B, C} *or* {B, C, D}. With out further information about whether smaller subsets of components being faulty would explain the fault, it is not possible to discriminate further.

Alternatively, from the point of view of looking at conflict sets, [9] [10] [68], {C} and {B} are both conflict sets. The minimal hitting sets of all of the conflict sets for a system are diagnoses, and the minimal hitting sets for {C} and {B} is simply {B,C}. Since only conflict sets that contain 1 component are being looked at, and not all of the conflict sets of the system, it is difficult to say with total confidence that {B,C} is the only diagnosis. The super sets of {B,C} could also be diagnoses i.e. the diagnoses for the system are {B, C}, {A, B, C} *or* {B, C, D} as before. Looking at table 6.4 the diagnosis that {A, B, C} are faulty could explain the behaviour. If the table showed that this super set of {B, C} could not explain the behaviour then it could have been eliminated from the possible diagnoses.

Because every conflict set cannot be evaluated, Reiters Hitting-Set Tree cannot be fully used, with its associated pruning rules. However the following rule can be applied, when all of the hitting sets from the limited number of conflicts sets have been gathered.

Once all of the hitting sets for this limited number of conflict sets have been gathered, then for any hitting set H_1, it must not have a proper subset, S, which has been shown <u>not</u> to be a conflict set, unless there is another hitting set H_2, such that $H_2 \cap S = \{\ \}$. If there is no other hitting set H_2, then the subset of components S can be removed from H_1, and the new H_1 will still be a diagnosis. To clarify, if

H_1 is a hitting set on the set of minimal conflict sets, then it is a diagnosis[68], i.e. if all of the components in the hitting set are assumed to be faulty, then this would explain the observations. However, if S is a subet of components assumed to be behaving correctly, and this can be shown to be consistent with the observations, then this situation can only occur if the remaining members of H_1 can explain the observations alone, *or* there is another set of components in another hitting set which when considered to be faulty are consistent with the observations and the assumption that the components in S are functioning correctly.

In summary, in the ideal situation, if a system has N components, N hypotheses must be tested. Each hypothesis is of the form { All the systems components except one} or from the conflict set point of view, one component is tested to see if when it is considered that to be behaving normally, it is in conflict with the observations. When these N hypotheses have been tested, any common subsets of any of the hypothesis which do explain the systems behaviour are found. These subsets could also explain the fault in the system. All of these hypothesis are then put forward as likely to be possible. Alternatively, using conflict sets a number of hitting sets on the conflict sets are generated. These are then checked as described above and then put forward as diagnoses. If there is more than one diagnosis, then the smaller ones should be checked first. It should be noted that it is by no means always possible to find just one likely diagnosis, more often than not there will be a number of diagnoses which are all correct, given that they are derived from just some of the systems measurements. In general, the larger the number of measurement points on the system, the fewer the number of likely diagnoses that will be generated, since more will be known about the system and it will be easier to discriminate between different hypotheses.

6.5 Limitations for Real Systems.

The method described above will only work in the ideal situation with real systems if there are enough measurement points on the system. In general, it is not possible to use hypothesis which say "All the components except one is at fault" nor to test a conflict set consisting of just one component. If this could be done, then it would mean the ability to test each component independently from all the others. If there are enough measurement locations on the system then this can be done but in general this will not be possible. As an example, consider a DC motor. A bond graph for this system is shown below.

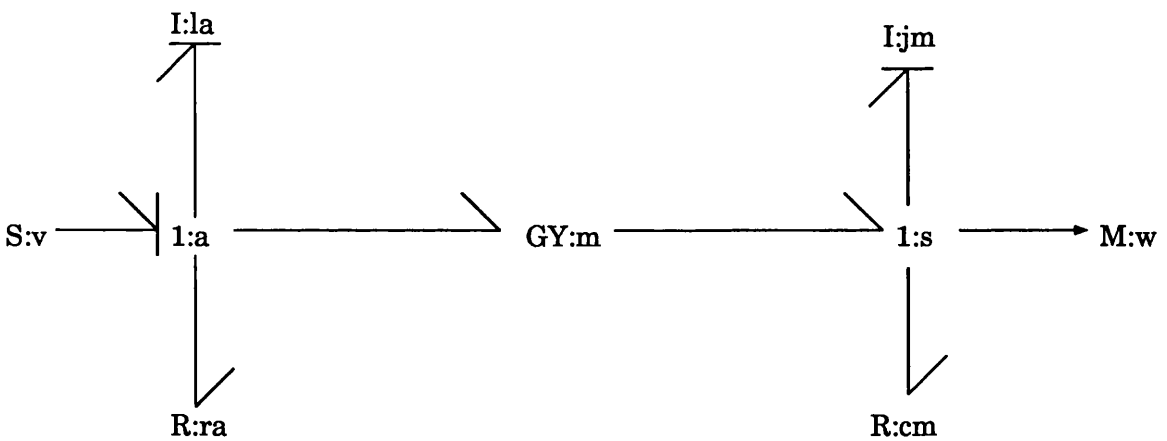


Figure 6.14: A bond graph of a DC motor.

This system has five components:-

- L, the inductance.
- R, the resistance.
- J, the inertia.
- D, the friction coefficient.

- Km, the conversion of electrical energy to mechanical energy and vice versa.

The transfer function of the speed of the motor, ω, in terms of the voltage, v is :-

$$\omega = \frac{K_m}{(K_m^2 + R.D) + (L.D + J.R)s + L.J.s^2}.V \tag{6.4}$$

which can be written as

$$\omega = \frac{1}{(A) + (B)s + (C)s^2}.V \tag{6.5}$$

where:-

$$A = \frac{K_m^2 + R.D}{K_m} \tag{6.6}$$

$$B = \frac{L.D + J.R}{K_m} \tag{6.7}$$

$$C = \frac{L.J}{K_m} \tag{6.8}$$

As was stated before, the DC motor has five physical parameters. Looking at equation 6.5, the dynamics of the system can be characterised by just three parameters, A, B and C. If the speed and the voltage of a DC motor were being measured, and a fault developed, then the fault will manifest itself as a change in one or more of the physical parameters. This will in turn change the values of the parameters in equation 6.5. If the dynamics of the system has changed, but not the structure of the system, then three suitable parameters for equation 6.5 can be found which will describe the new dynamics. Looking at equations 6.6 to 6.8, if A, B and C are known, any two of the physical parameter can be arbitrarily given any value, and the values of the remaining three physical parameters can be fixed. This results in the position that if faults occur in three or more of the physical components then there will be no way to differentiate between the possibilities.

If a hypothesis which has three or more components is tested, a positive answer will always be obtained because the transfer function has three components but the

system model has five physical components. So, testing for faults in three or more components simultaneously is pointless as a positive result will always be obtained. Therefore only hypotheses with one or two components can be tested on this system. This also means that with this system a conflict set can only be tested which has three or four components and no less. All the hypotheses that can be tested are listed in table 6.5, the potential conflict sets which may be usefully tested are shown in table 6.6.

1. {R}	6. {R, L}	11. {L, Km}
2. {L}	7. {R, D}	12. {L, J}
3. {D}	8. {R, Km}	13. {D, Km}
4. {Km}	9. {R, J}	14. {D, J}
5. {J}	10.{L, D}	15. {Km, J}

Table 6.5: All the hypotheses that are testable.

1. {L,D,Km,J}	6. {D,Km,J}	11. {R,D,J}
2. {R,D,Km,J}	7. {L,Km,J}	12. {R,D,Km}
3. {R,L,Km,J}	8. {L,D,J}	13. {R,L,J}
4. {R,L,D,J}	9. {L,D,Km}	14. {R,L,Km}
5. {R,L,D,Km}	10.{R,Km,J}	15. {R,L,D}

Table 6.6: All potential conflict sets that are testable.

There are fifteen potential conflict sets which can be differentiated between. Using the method of examining subsets as described in the previous sections, this can be reduced to ten, i.e. sets 6 to 15 in table 6.6, which is equivalent to the potential conflict sets 6 to 15 in table 6.6. This is obviously not as good as reducing the number down to five which is the ideal minimum, but this is the minimum number of hypotheses which need to be tested as the result of any hypothesis with more than two components is meaningless.

Although this is not a significant improvement in reducing the number of hypotheses (reduced from 15 to 10), using only 1 or 2 component failures, consider the implications of measuring the current as well as the speed and voltage. If this was done, then all five physical parameters could be identified uniquely for any new behaviour of the motor, i.e. there is only one solution if it were assumed that all the physical parameters are unknown. It is now possible to differentiate between 1, 2, 3, 4 or even five components failing. Using the method described in the previous sections of this chapter, one can reduce the number of potential conflict needed to test to just five and one can find which of the 30 hypotheses are most likely to be correct.

In summary, the limitations of this method depend heavily on the system being monitored and the number and location of the measurements which are made on each observation of the system. Even with these limitations, it is still possible to reduce the number of hypotheses which need to be tested.

Doing this in the non-ideal situation may mean that some of the diagnoses which arise are not minimal diagnoses, however all diagnoses will be valid. Reducing the number of conflict sets which need to be tested means that it becomes possible to produce real time diagnosis for systems with a large number of components: the price to pay is that some of the diagnoses will not be minimal. The number of non-minimal diagnoses will be reduced as the number of sensor on a system is increased. Also, due to the limitations on identifiability of real systems described above, any multiple fault hypothesis larger than the biggest identifiable one can be discarded. The result is that in many real situations, reducing the number of conflict sets to test can mean a diagnostic system which will work in real time and one which does produce meaningful and useful results.

6.6 Finding the Most Likely Cause of a Fault

This section discusses how to find the most likely hypotheses, out of all of the multiple and single fault combinations. In general, it is not possible to find a single hypothesis with which it can be said with confidence, that it is the cause of the fault. Instead, a list is produced of all the hypotheses which do explain the fault and then this is ordered according to some ranking scheme, to give some indication of how well each one does actually matches the systems' behaviour.

According to Reiter [68], a correct diagnosis is a hitting set on the conflict sets. Since it cannot be said with absolute certainty which sets are conflict sets (due to noise and model uncertainty), this definition of a correct hypothesis must be adapted. As discussed earlier a conflict set is given an error index which indicates how much it is in error with the systems' behaviour. If the error index is low (close to the noise/signal ratio), then the set is not in conflict with the systems' behaviour. If the error index is high (a value of 1 is very high), then the set is in conflict and it is a "conflict set". A correct hypothesis is now, therefore, a set which hits all of the potential conflict sets which have a high error index, and doesn't hit those with a small error index. Determining what is a high error index and a low error index is not difficult, but when an error index is in between high and low then things become unclear as to whether it is a conflict set or not. This depends upon the noise content of measurements and how accurate the original model is. A simple threshold approach would be unsatisfactory in these situations.

To solve this problem another index is calculated which indicates how well a hitting set hits those conflict sets with high error indices, and how well it misses those with low error indices. This will be called the "hitting index".

Firstly it is noted that if an error index has a value of 1, then it indicates that the estimates of the outputs are approximately 100% in error when compared to the

actual outputs. Such a set would clearly be a conflict set. Also, it is possible for the error index to be above 1. If the error index is 0.5, this indicates a 50% error. This is also clearly an indication of a conflict set as a 50% error is substantial. As the error index decreases to zero it becomes less clear whether a set is a conflict set, until the error index approaches the noise/signal ratio on the system measurements. The first step in calculating the hitting index will be to limit the maximum value for an error index to be 0.5, i.e. if it is calculated to be greater than 0.5 then it is set to 0.5. The hitting index is now defined to be Σ *(0.5 - the error indices for the conflict sets it hits)* + Σ *(the error index for each of the conflict set it doesn't hit).*

The first summation, Σ *(0.5 - the error indices for the conflict sets it hits)* will be smallest when the sets that the diagnosis hits have large values for their error indices, and it will be largest when the sets the diagnosis hits have small values for their error indices. The second summation, Σ *(the error index for each of the conflict set it doesn't hit)*, will be smallest when the sets the diagnosis doesn't hit have small error indices. It will be largest when the error indices of the sets the diagnosis doesn't hit are large (remember the largest value for an error index is 0.5).

So, if the diagnosis hits all the conflict sets which have large error indices and misses those with small indices, it will have a small hitting index. If it hits all the ones with small indices, and misses those with large indices, it will have a high hitting index. If it hits and misses a mixture of high and low error indices then it will have a value in between these two extremes.

The most likely diagnosis will therefore be the one with the lowest hitting index. As an example, in table 6.7 below the results for a typical diagnosis on a system is shown. Here only the top eight results are shown from another example system which has seven components, labeled A to G. The hypotheses are listed in the middle column, and they are ordered according to their hitting index which appears in column

three. It should be noted that it is not possible to just take the top result or the top n results and expect the actual fault to always be there; as can be seen from the table, the first four results are all reasonable representations of the fault, the 5th and 6th results are also not out of the question. The list represents those hypotheses which should be checked first.

Ranking	Hypothesis	Closeness of Behaviour Match
1	{A, C}	3.96
2	{A, G}	3.98
3	{A, B, C}	4.12
4	{A, F, G}	4.18
5	{A, C, E}	4.59
6	{A, F, G}	4.65
7	{D, E}	5.45
8	{A, B, E, F}	5.82

Table 6.7: Typical Results during diagnosis.

An outline will now be given of the algorithm which is used to generate the hitting indices for each hypotheses when only testing a limited number of potential conflict sets.

This begins with all of the conflict sets which have been tested, and for each of these this is a value which indicates how well it fits the systems' faulty behaviour. These conflict sets will consist of as few components as possible, but not less than 1 component, (this is the ideal situation, often it would not be possible to test conflict sets of just 1 component). So for example, in the previous section a DC motor was discussed with measurements of v and ω, in this case, the starting point is 10 conflict sets which had been tested and each one of them would be made up of three components which are assumed not to be faulty. If v, i (the current) and ω were being measured, then there would be five conflicts sets and each one of them would be made up of one component which is assumed to be not faulty. If all the conflict sets which have 1 component are actually tested, then this is the best situation as the minimum

number of conflict sets N are being tested, and the hitting indices for all the possible hypotheses can be calculated. If, as in the DC motor example with measurements of v and ω only conflict sets with more than 1 component can be tested, then it is necessary to test more than just the minimum N conflict sets, but the number tested will still be less than $2^N - 1$, which is the maximum number of conflict sets which would normally need to be tested as discussed in section 3.10.

Difficulties can arise when it is not possible to test all of the conflict sets of a given size, because of either the structure of the model and/or the equations needed to test a conflict set with that number of components cannot be solved uniquely. For example, suppose a system which has five components is being considered, and one starts by looking at those conflict sets which contain 1 component. This requires the ability to estimate the parameters of all of the other components in the system and then see which of these have a unique solution, as these are the ones which need to be tested. If all of them can be tested, then there is no problem and the procedure described above is followed. If it is not possible to test any of them, then there is also no problem yet, the number of components, in each conflict set is increased to 2 and then these are checked for unique solutions. The problem arises when some of the 1 component conflict sets can be tested and some cannot.

To explain further, if all of the conflict sets with 1 component can be tested, then these can be used to calculate hitting indices for all of the possible hypotheses, and these will result in a set of minimal diagnoses. If none of the equations can be solved for the 1 component conflict sets then the 2 component conflict sets must be checked, then 3 components etc. If its not possible to solve any equations when looking at N-1 components, but it is possible to solve some, although not all of the equations at N components, then it may well be the situation that there is no information available for some of the hitting sets, it will certainly be the situation that there is

more information about certain hypotheses than others. In this position the error indices for the N+1 components should also be calculated and checked to ensure that all of the equations needed to calculate the parameters of the components, excluded from each conflict set, can be solved.

The number checked will still be less than the total $2^N - 1$ possible conflict sets but it will be necessary to check more than the minimum N conflict sets.

6.7 Summary

In this chapter it has been shown that when the number of sensors on a system are limited, then it is possible to deduce more about the system if it is considered that the fault has a constant behaviour during the diagnosis process. A conflict set is therefore not just a subset of components which, when assumed to be behaving normally, are consistent with the observations. But it is a subset of all of the systems components which, when assumed to be behaving normally, are consistent with the observations, *AND* those components which are not in the subset are exhibiting a consistent behaviour for every observation.

When noise is present checks must be made to ensure that the behaviour of components which are suspected to be faulty are constant by using their new behaviour in the model of the system and estimating what the outputs of the system should be, given these new behaviours. By comparing the estimated and actual outputs one can determine whether faults in these components are consistent with the observations, and hence whether the other components (those assumed to be behaving normally) can be considered to be a conflict set.

The number of subsets of components for a large system is very large indeed. If every subset was checked to see if it was a conflict set, the amount of computation required would be enormous, and on-line, real time diagnosis would not be possible

except with small systems. To overcome this practical implementational problem, only a small number of sets are checked. These will be those potential conflict sets which have as few members as possible, (but at least one member), and which allow the identification all of the other component parameters of the system which are not members of the conflict set.

When noise is present, nothing is certain, so an index is assigned to each subset of components tested which represents, in some sense, how likely it is that the set is a conflict set. This index is then used to find out how well a diagnosis (hitting set), hits those subsets which are likely to be conflict sets, and misses those subsets which are not likely to be conflict sets. A "hitting index" for each diagnosis is calculated. All of the hypotheses are listed, together with their hitting indices in order of the hitting indices. Thus the diagnosis at the top of the list is considered to be the most likely, and any others with similar values for their hitting index are also quite possible.

For every sample of data taken from the system, the parameters of components considered to be faulty will be re-estimated. Then the system outputs are re-estimated for each conflict set, the error indices are re-evaluated for each conflict set, and finally the hitting indices are re-calculated for each diagnosis. The results of this will be updated for each time interval. At the start of diagnosis, it is likely that the list for the most likely diagnoses will change rapidly, but as parameter estimates become more stable, so will the systems' output estimates, the error indices and the hitting indices. The longer that is spent diagnosing the system, the more stable that the list of the most likely hypothesis will become. After sufficient time those diagnoses reported as being the most likely can be checked.

Chapter 7

Experimental Results.

In this chapter, the techniques described earlier will be applied to a real system. It will be shown that these methods can successfully detect a fault occurring, identify the component which is faulty, and quantify the size of the fault even in the face of modelling errors and noisy measurements. The actual physical size of the system under test adds confidence that there are many real situations in which this approach to fault diagnosis will work in practice.

7.1 Modelling the System.

The system that is used here consists of a water tank,a pump, and an arrangement of pipes, valves and sensors. A sketch of this system is shown in figure 7.1. The actual physical size of the system is approximately 3.5 metres in height and 2 metres by 2 metres in length and breadth.

As can be seen in the sketch, water is pumped from the sump, through a flow meter and the input valve into the tank. Water in the tank flows through the output valve back to the sump. The depth of the water in the tank is measured by a sensor which measures the pressure at the bottom of the tank. The input flow sensor measures the

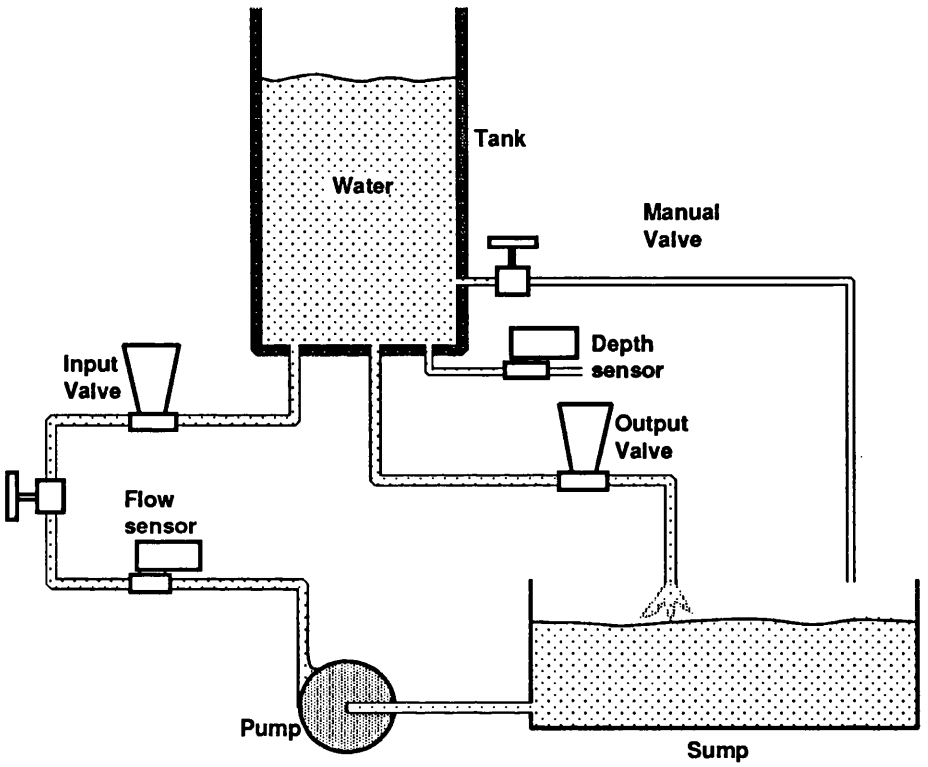


Figure 7.1: A sketch of the system to be tested.

pressure drop across an orifice. These two signals can be read by a computer at any time. The input valve and output valve are controlled using compressed air and can be adjusted via a computer. The valve openings are measured in percent, 100% is fully open and 0% is fully closed. The pump is running continuously, directly from the mains power supply.

In these tests, a computer is used to control the position of the input and output valve openings, according to the level of the water in the tank, as if it were following some process recipe. An example recipe is shown in table 7.1.

This is obviously a fairly straight forward control strategy, but the complexity of the systems' control is unimportant as far as detecting and diagnosing faults in the system itself are concerned. As long as the system is being stimulated then measurements can be used from the system to determine whether it is behaving correctly.

Step	Action
1	Close output Valve
2	Open input Valve
3	Wait for level to reach 1.2 metres
4	Move input valve to 30% open
5	Move output valve to 75% open
6	Wait for level to drop to 0.3 metres
.	etc

Table 7.1: An example recipe.

The first computer controls the system and changes its' inputs, this causes the systems' outputs to change. The inputs, as set by the first computer and the measurements from the systems' outputs are sent to a second computer which contains the on-line fault detection and diagnosis software. An overview of this set up is shown in figure 7.2.

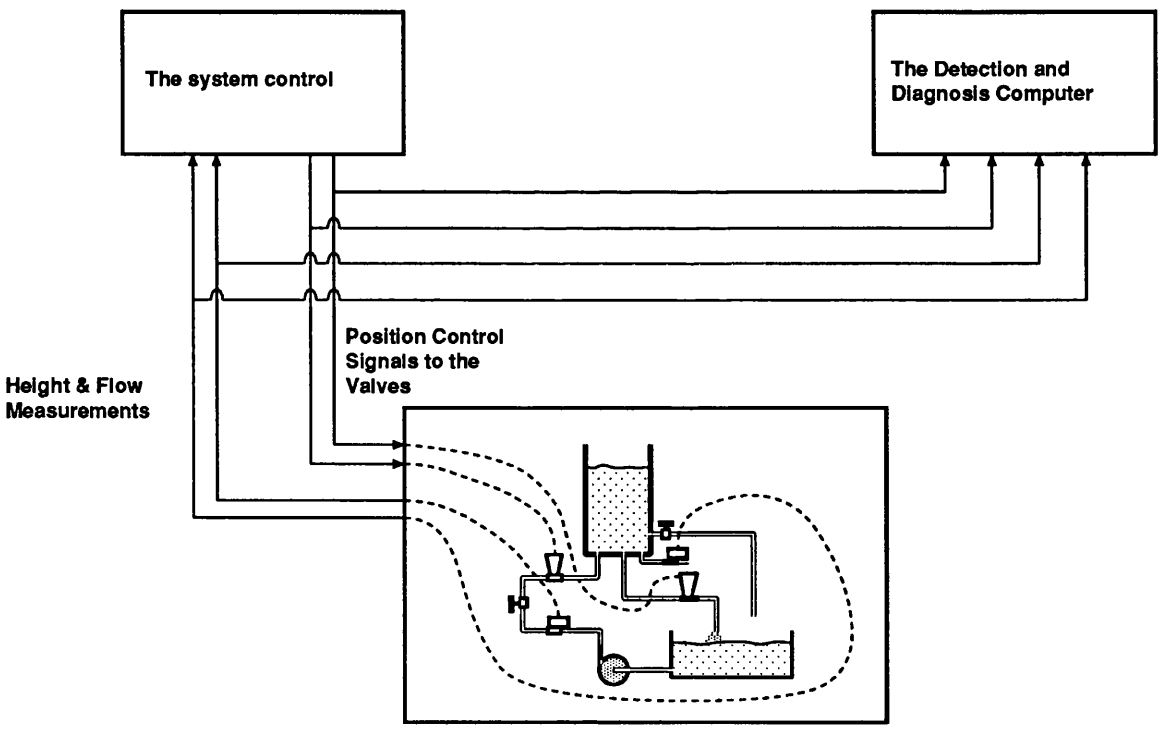


Figure 7.2: The set up of the system.

A block diagram and a bond graph for the system are shown in figure 7.3. This model for the system was arrived at by carrying out a number of different tests on

the system, and a brief description of the models' parameters is given below.

- K_1 represents the constant of proportionality between the square of the flow into the tank and the pressure loss due to friction through the pipes.

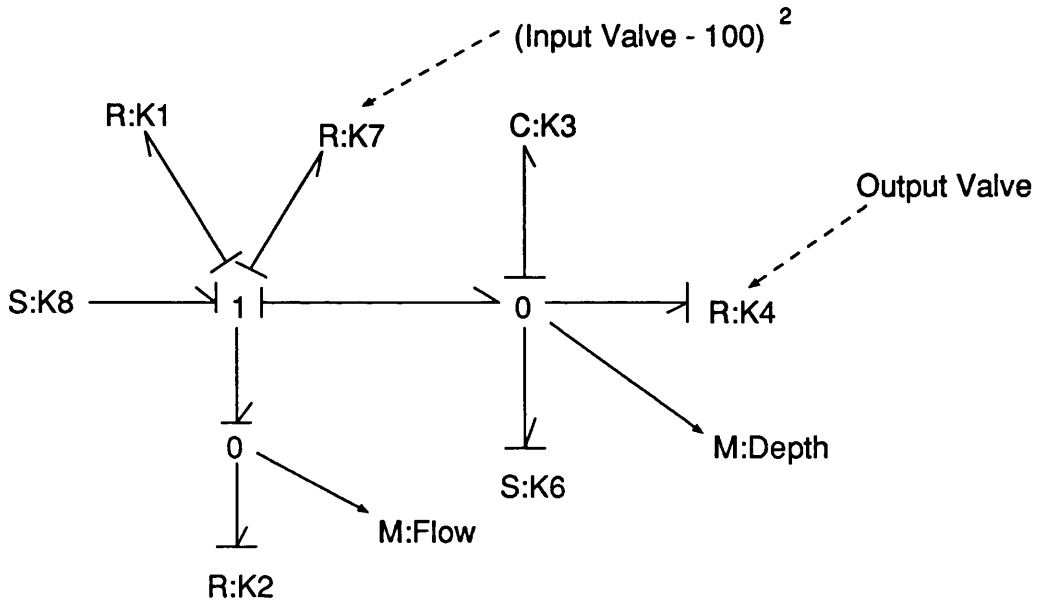
- K_2 represents the constant of proportionality between the pressure drop measured by the flow meter and the square of the flow in m^3/s. e.g. $Q_{in} = \sqrt{\frac{dp}{K_2}}$, where $K_2 =$ (Area of orifice $\times$ Coefficient of discharge for orifice)$^{-2}$.

- K_3 represents the constant of proportionality between the volume of water in the tank and the pressure at the bottom of the tank. $K_3 = \frac{\rho \times g}{Area}$.

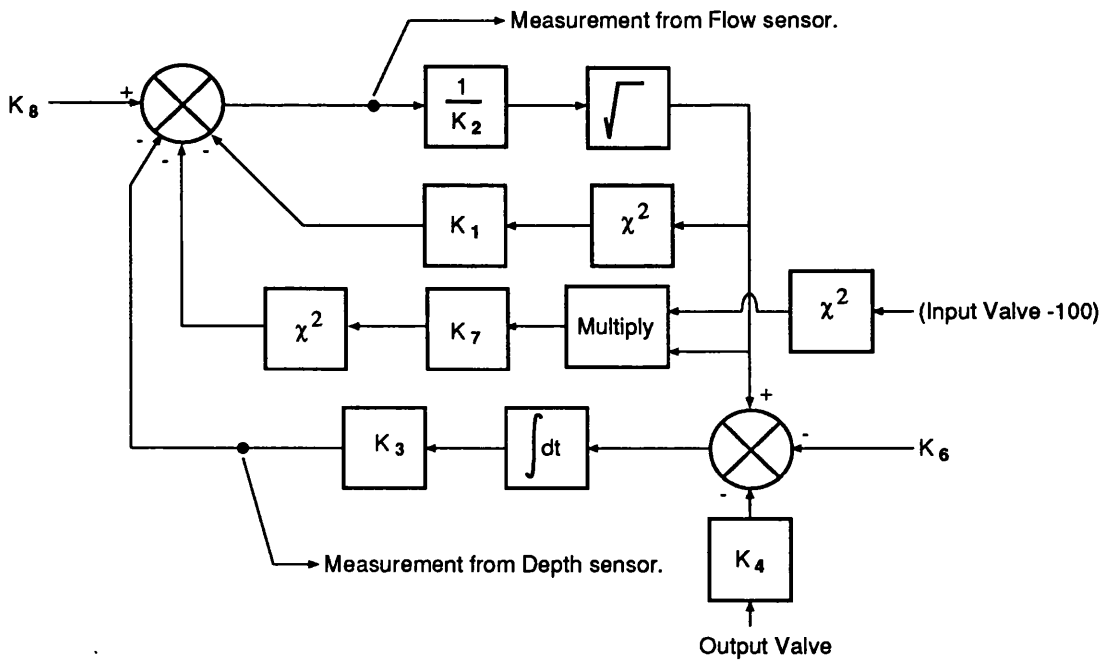
- K_4 & K_6 are used to parameterise the flow of water out of the tank. This was found be very nearly linearly related to the output valve's opening. $Q_{out} \approx K_4 \times Output\ valve\ position + K_6$. This can be seen in figure 7.4. Here it is seen how the level of water in the tank drops when the input valve is closed and the output valve is in different openings. (100% means fully open.) The rate of change of level remains essentially constant as the level drops. These results are shown in figure 7.5 where it can be seen that when the output valve is between 10% and 65% the flow out is approximately a linear function of the valve position.

- K_7 is used as a parameter to the input valve. The actual constitutive relationship for the pressure drop across the valve for any particular flow and valve position was not available, however an approximation to it was found using the results of a number of test. $Pressure\ drop \approx (K_7 \times (100 - Input\ Valve)^2 \times Q_{in})^2$.

- K_8 is used to represent the pressure from the pump. For the system, this can be regarded as being virtually constant. The specifications for the pump indicated that at flow rates lower than 16 l/sec, the pressure was constant.



Bond graph of the system.



Block diagram of the system.

Figure 7.3: The model of the system.

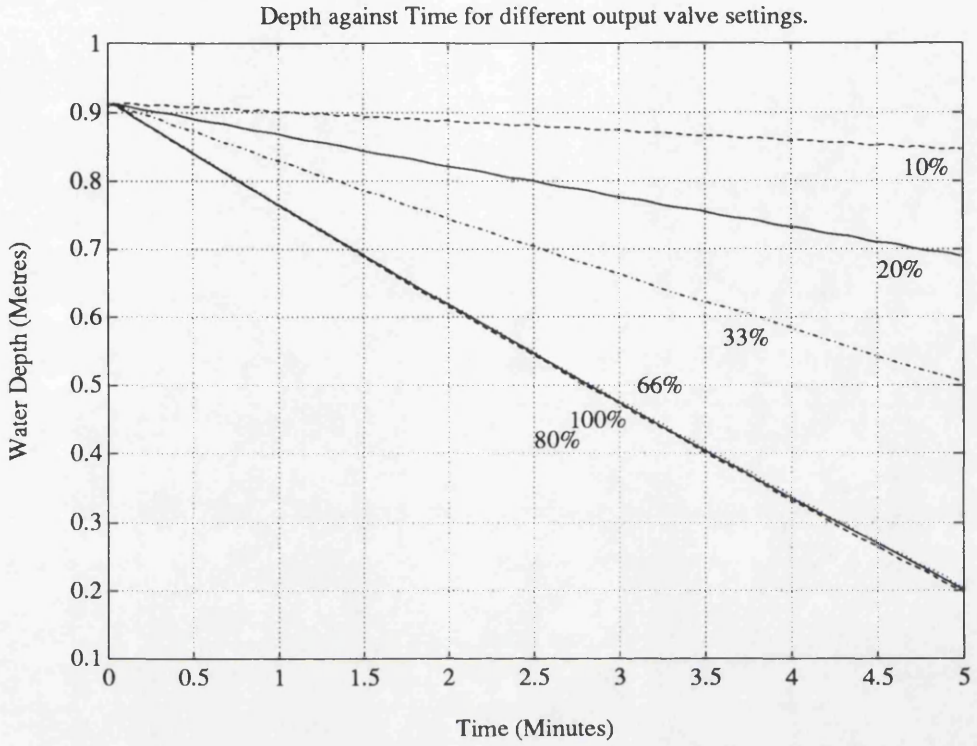


Figure 7.4: Testing the output valves' characteristics.

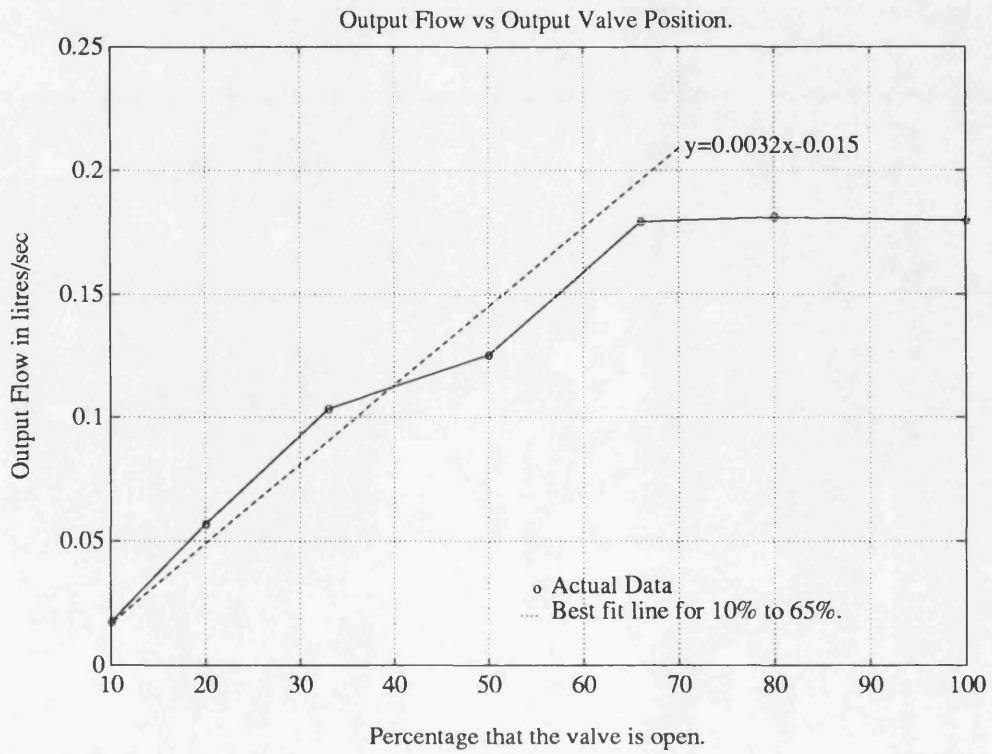


Figure 7.5: The output valve characteristics.

The values obtained for the parameters are shown below in table 7.2. Pressure is measured in Pascals, and the flow is measured in m^3/sec. This is the model of the system. The major relevant physical components of the system are modelled and have a parameter associated with them. Our model does involve a number of approximations, both in parameter values and the constitutive relationships of the components, but one of the objects for the detection and diagnosis system was that it should be able to cope with model inaccuracies, as they will always be present in any system model.

Parameter	Value
K_1	2.41×10^{11}
K_2	5.08×10^{11}
K_3	1.30×10^{5}
K_4	2.35×10^{-6}
K_6	1.04×10^{-5}
K_7	255
K_8	64650

Table 7.2: The parameter values.

7.2 Detecting Faults.

In this section the performance of the fault detection part of the algorithm in three different circumstances will be examined. Firstly in the case when no fault occurs, the objective here is to ensure that no fault is detected bearing in mind that the model of the system is not perfect and there is noise in the measurements taken from the system. Secondly in the case when a restriction is imposed in the pipe between the pump and the tank. Thirdly the performance of the detection process will be examined when a leak occurs in the tank.

Here only the detection of fault is looked at, not with the diagnosis of the fault. This will be looked at in the next section.

7.2.1 Monitoring the System without a Fault.

Figures 7.6.A and 7.6.B show the signals sent to the input and output valves by the control computer. Figures 7.6.C and 7.6.D show the signals returned by the flow and height sensors. Samples are taken once every second and are passed to the detection/diagnosis computer.

The algorithm was told that a 2% noise/signal ratio was expected and that the model was 95% accurate. The signals from the input and output valves were not filtered at all. The signals from the flow and depth sensors were filtered using a sixteen point best fit parabola as described in chapter 4, section 6.

As described earlier, the fault detection algorithm works by finding out what it expects the outputs of the system to be and then comparing these to the actual outputs of the system. An error index is then calculated based upon how different the expected and actual outputs are. (See chapter 5, section 4.)

Figure 7.7 shows the signal returned from the flow meter (the dashed line) and the expected value for the flow reading based upon the system model and the inputs. Also shown (labelled "Actual error") is the difference between these outputs. The expected flow signal and actual flow signal are very similar. The largest error occurs near the beginning when the input and output valves openings are gradually changing (see figure 7.6). The error here is larger because, as stated earlier, the constitutive relationship for the input valve was not known and an approximation was used to describe its' behaviour.

Figure 7.8 shows the signal returned from the depth meter. The difference between the actual signal and the expected signal is very small. These two figures show that the model of the system is fairly close to the real systems' behaviour.

Finally, in figure 7.9, the value for the error index is seen, this was calculated as the data was read from the system. The error index is large around the 5 minute mark

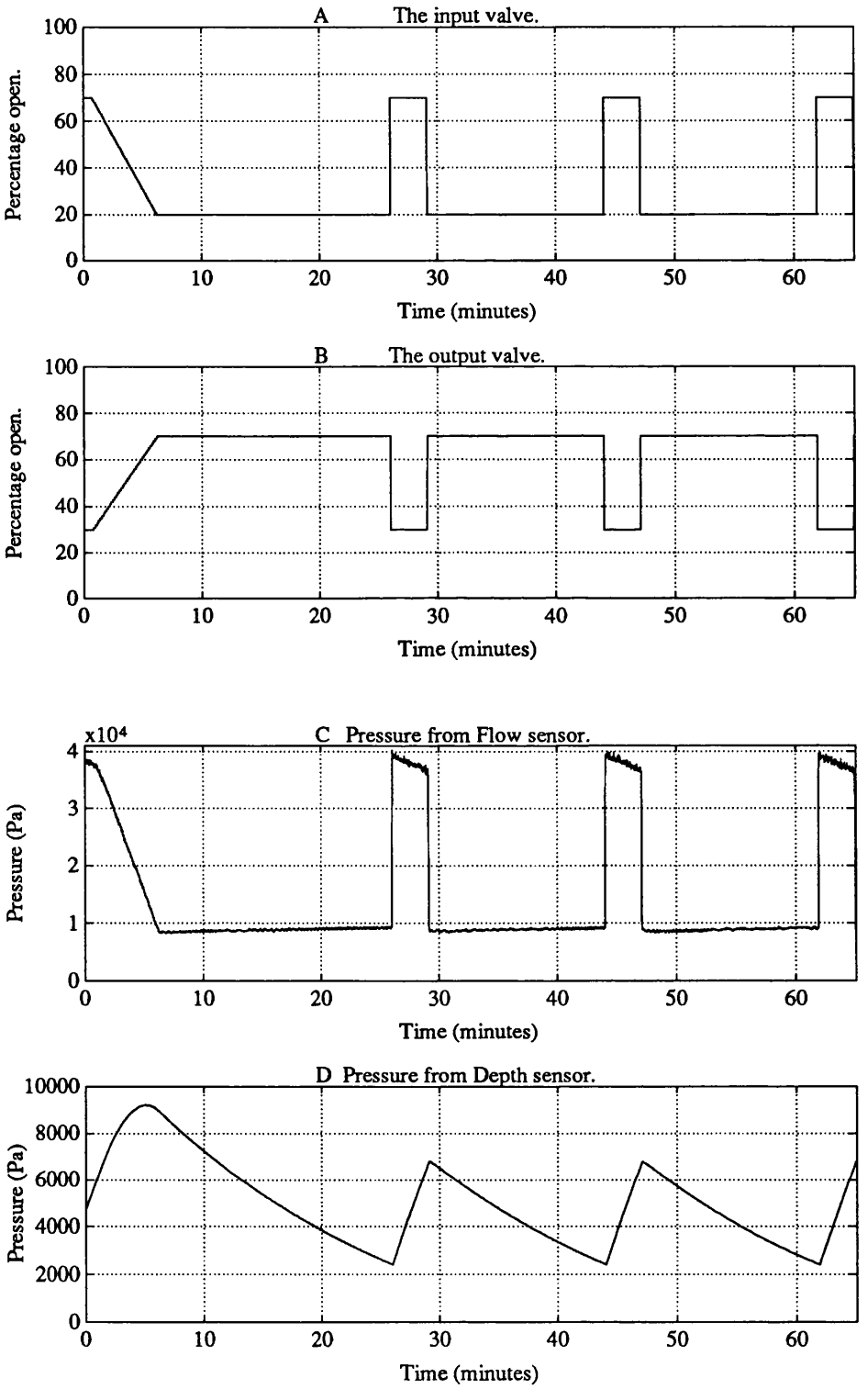


Figure 7.6: Measurement taken from the system.

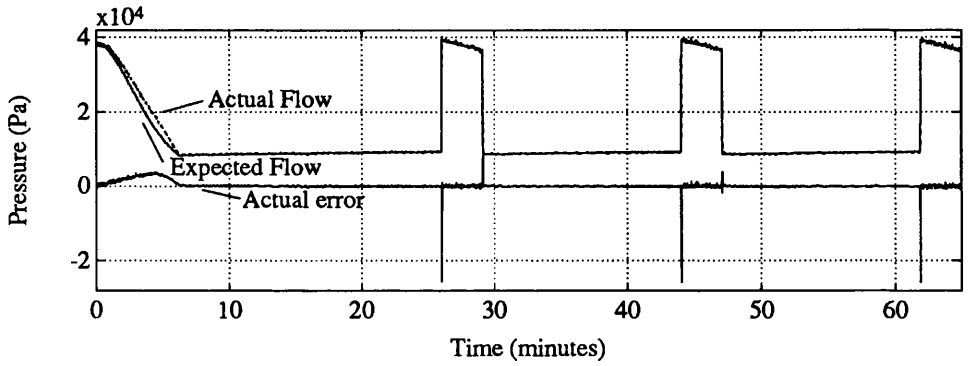


Figure 7.7: Comparing the expected and actual flow.

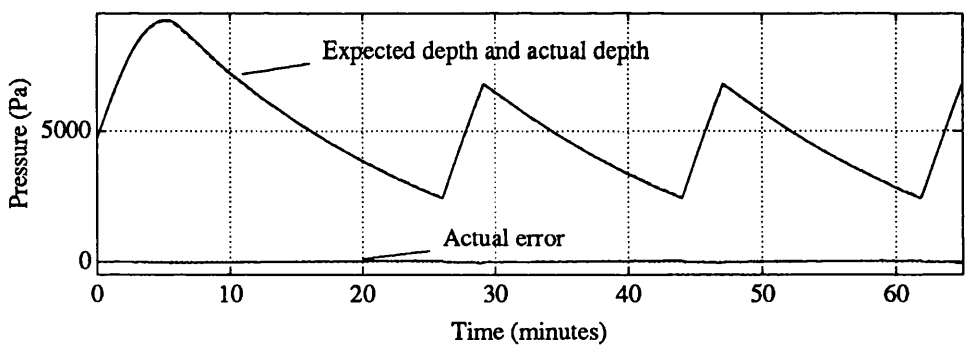


Figure 7.8: Comparing the expected and actual depth.

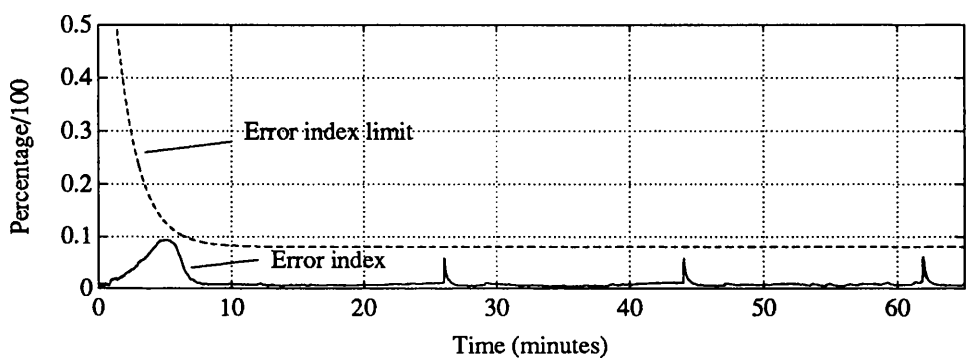


Figure 7.9: The error index and its' limit.

because of the error in the flow signal seen in figure 7.7. The spikes at 27 minutes, 44 minutes and 62 minutes are caused by the step changes in the valve openings, although these do not exceed the error index limit, they would not cause a fault to be detected even if they did, as was explained in chapter 5, section 4, page 110.

If it were found that the error index did exceed the error index limit because of the error in the flow signals, then it would be necessary to decrease the model accuracy value given to the algorithm. In this case the model was stated to be 95% accurate, decreasing this value would have the effect of reducing the size of the flow error and therefore the error index, but it would also mean that smaller faults would not be detected.

It has just been just shown that when the monitoring process on the system is run where no fault is present, no fault was detected. This is in spite of modelling errors and noisy measurements.

7.2.2 Detecting a Fault Occurring.

Having shown that when there is no fault in the system, no fault is detected, the next step is to show that when a fault is present, it is detected. To show that the detection method described does work on real systems, it will be demonstrated with two examples. The first one is a blockage occurring in the pipe, and the second one is a leak appearing in the tank.

Example 1. Blockage in pipe.

The blockage was induced by slightly closing a manually operated valve on the pipe between the pump and the tank. This can be thought of as a slight blockage occurring in the pipe. The result of this is that the parameter associated with the friction in the pipes (K_1) effectively increases. This change in the systems behaviour will cause

a larger error when compared to the expected systems behaviour, and hence a fault will be indicated.

Figure 7.10 shows the input and output valve signals, and figure 7.11 shows the measured flow and depth signals. The "fault" occurred at time 41 minutes. A small spike can be seen on the flow signal at this time.

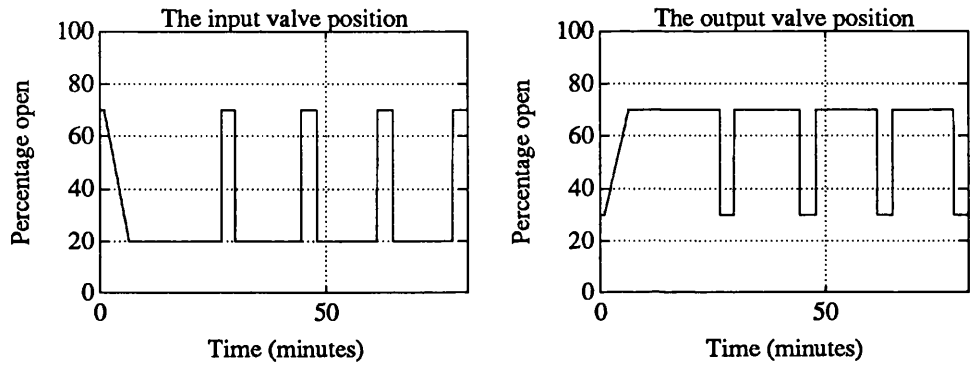


Figure 7.10: The input and output valve signals.

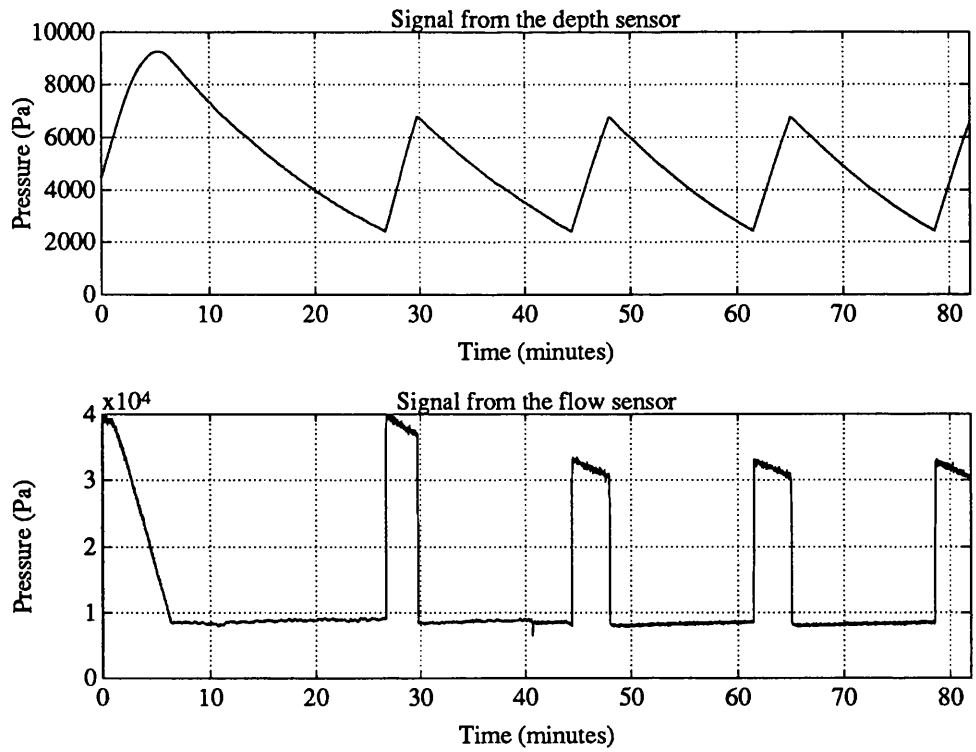


Figure 7.11: The flow and depth measurements.

Figure 7.12 shows the actual signal from the flow sensor and the expected value.

After time $\approx$ 41 minutes the difference between the two signals is substantial. Figure 7.13 shows the expected and actual signals from the depth sensor. Here the difference is small. This is because the **actual** value for the flow is used in calculating the expected depth, and doing this does not involve using the pipe friction component, so the expected depth is essentially correct. The calculation for the flow does however need to use the pipe friction component[1] and it will therefore not be the same as the actual flow since the parameter for this component has changed in the system.

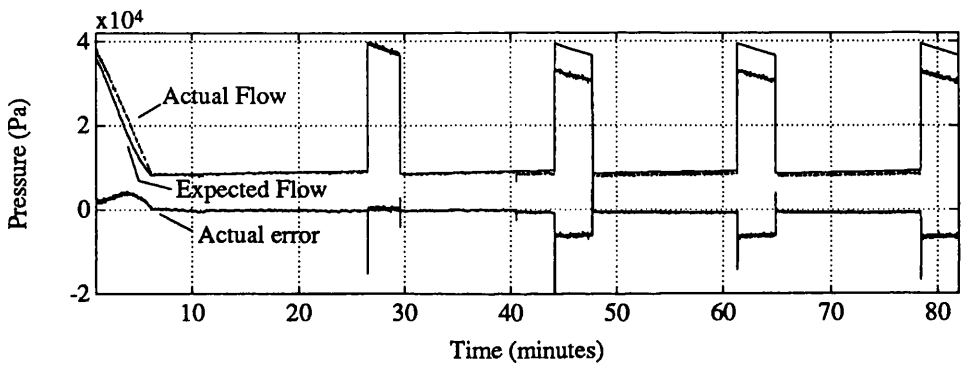


Figure 7.12: The actual and expected flow.

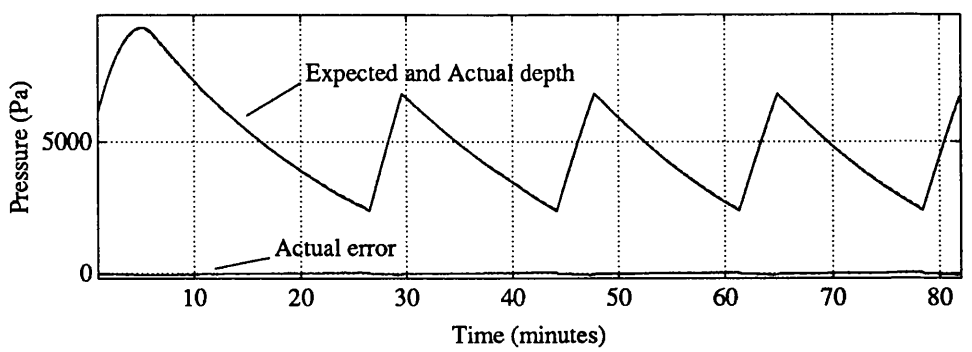


Figure 7.13: The actual and expected depth.

Figure 7.14 show the error index and its' limit. At first this is quite similar to figure 7.9, that is up until the fault occurs. Immediately after the fault has occurred,

[1]Because the outputs are calculated causally, the signal obtained by differentiating the depth cannot be used to help calculate the flow, so the pipe friction component has to be involved in the flow calculation. (See section 4.2.4)

the error index does not change greatly, as the actual and expected flow do not differ greatly. However, when the flow starts to increase, the difference between the actual and expected flow increases quickly, as does the error index. At time $\approx$ 44 minutes 30 seconds, a fault is detect. The error index is larger than the error index limit, and it is not decreasing. Normally, the fault detection process ends here and diagnosis begins, but in figure 7.14 the detection algorithm has been continued so that the behaviour of the error index can be observed.

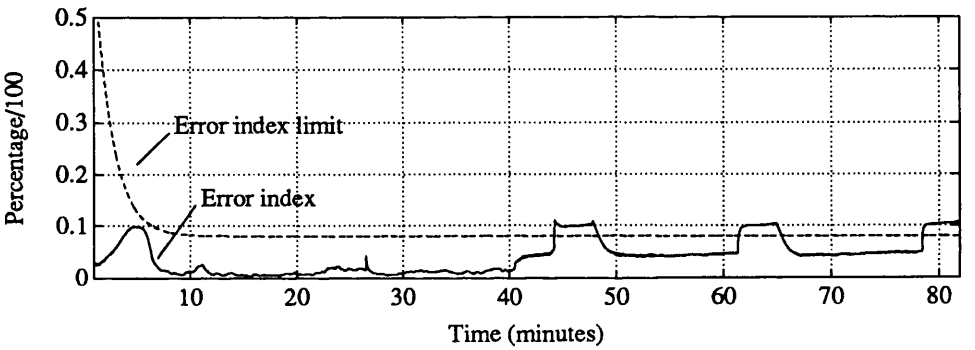


Figure 7.14: The error index.

Example 2. A leak in the tank.

The leak was induced by slightly opening a manually operated valve on the side of the tank. This allowed water to flow directly back to the sump. The result of this is that the flow out of the tank will be larger than that expected according to the output valve opening. As before, the change in the systems' behaviour will cause a larger error when compared to the expected systems' behaviour, and again a fault will be indicated.

Figure 7.15 shows the input and output valve signals, and figure 7.16 shows the measured flow and depth signals. The "fault" occurred at time 10 minutes.

Figure 7.17 shows the actual signals from the flow sensor and the expected value. After time $\approx$ 10 minutes the difference between the two signals is still very small.

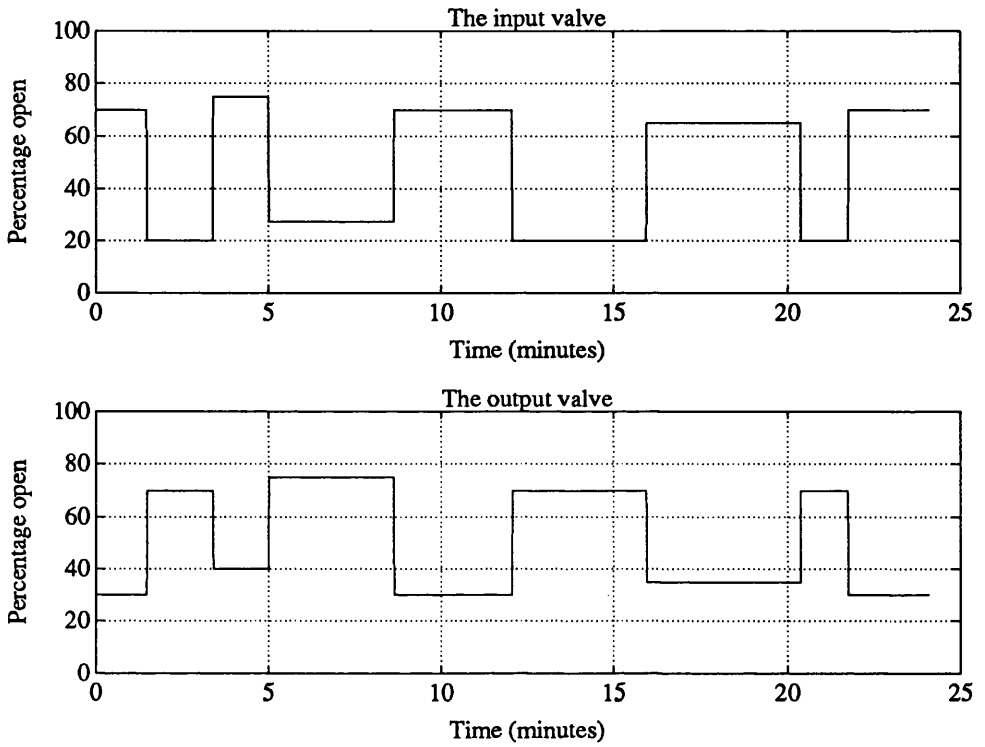


Figure 7.15: The input and output valve signals.

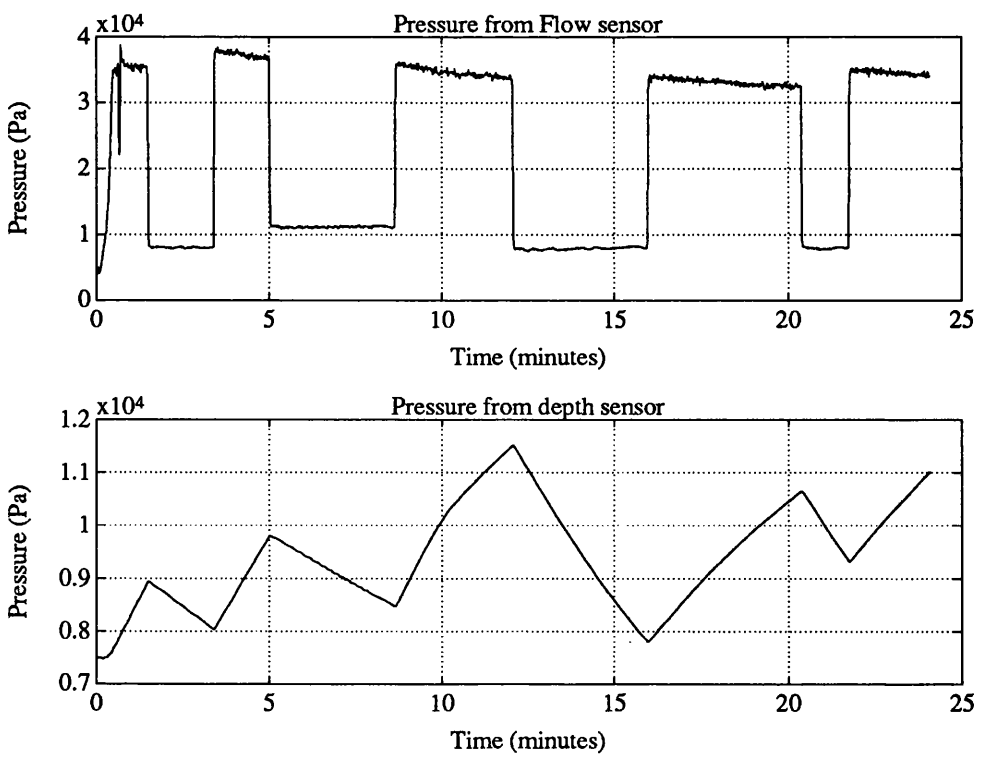


Figure 7.16: The flow and depth measurements.

Figure 7.18 shows the expected and actual signals from the depth sensor. Here the difference is significant after 10 seconds. As before there is an error in one of the predicted outputs, but not in the other one. This is because the predicted value for the flow uses the measured value for the depth of water in the tank (rather than a predicted value) and the characteristics of the output valve for the tank occur, in a causal sense, after the flow meter. (The fault is in the output valve from the tank.) Therefore the predicted value for the flow will not be affected by this fault.

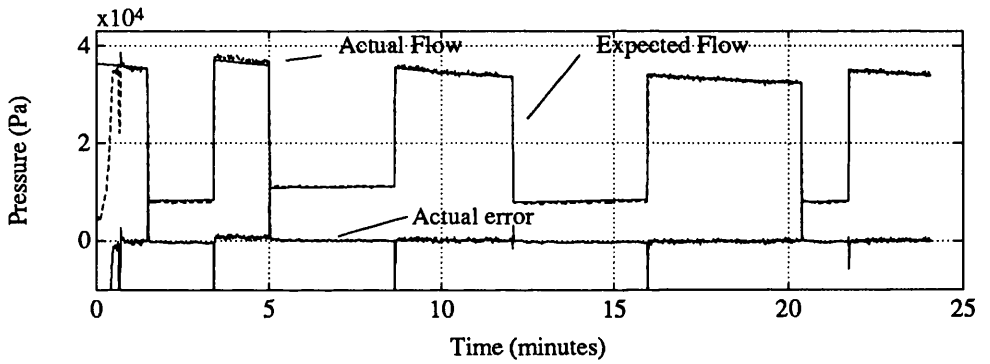


Figure 7.17: The actual and expected flow.

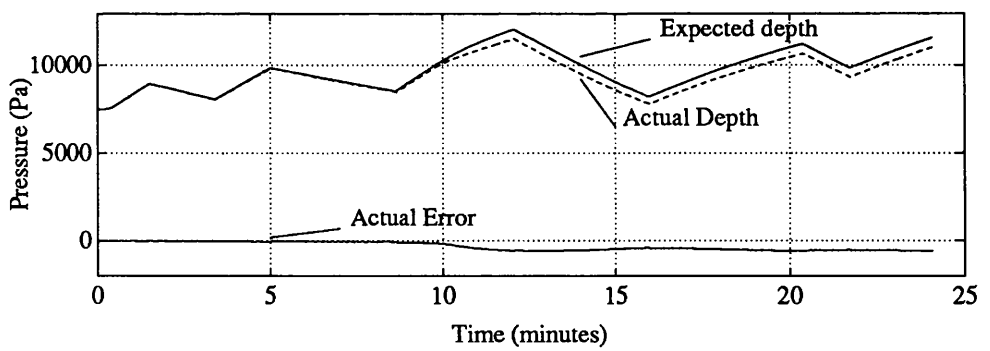


Figure 7.18: The actual and expected depth.

Figure 7.19 show the error index and its' limit. At first the error index is lower than its' limit. At time $\approx$ 10 minutes the error index can be seen to gradually increase until it exceeds the error index limit at time $\approx 12\frac{1}{2}$ minutes. As before, the fault detection process would normally begin at this time but the detection algorithm has

been continued again so that the behaviour of this error index can be observed. It is seen to remain high at about the same level as the error index limit. Obviously, if the leak had been larger, then the error index would have risen more rapidly, and it would have settled at a higher level.

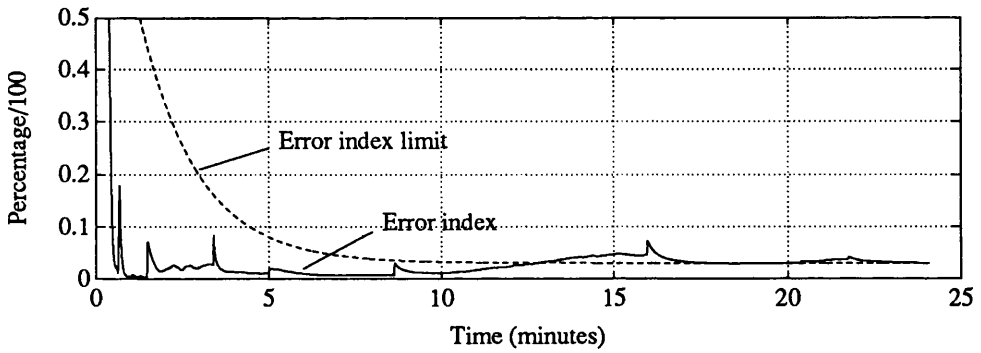


Figure 7.19: The error index.

Summary of detection results.

It has been shown that when no fault was present, no fault was detected. The error index remained small except for some sharp rises due to step changes on the inputs. This occurs when there is a step change in the inputs to the system which occurred between measurements. For example, if measurements were being made once every minute, and the inputs to the system changed five seconds after one set of measurements, then the system will have been changing for 55 seconds before the diagnosis system realises that the input have changed. This means the states in the model will have been calculated assuming the wrong set of input values for the last 55 seconds. In the examples this did not cause the error index to exceed its' limit, but even if it had it would not have triggered a fault as having been detected. This is because immediately after the sharp rise in the error index, its' subsequent values quickly dropped to levels significantly less than the error index limit. A fault is only detected if the error index limit is exceeded, and the error index is not decreasing.

This prevents false alarms due to step input changes.

The error index limit is based upon the expected noise content of the measured signals. The higher the noise, the higher limit.

The value given for the accuracy of the model of the system affects how quickly the algorithm can modify the models states so that the models' output reflects the measured output of the system. If the accuracy is 100% then the models' outputs will be based solely on the system inputs and the previous states of the model. Any small error in the system model will result in an increasing error between the systems' and the models' outputs, and in this case false alarms would occur very frequently as the model can never be perfect.

If the accuracy is 0%, then the outputs of the model will be based solely on the measured outputs of the system. i.e. the predicted and actual outputs will always be exactly the same, no matter how inaccurate the model is, also faults will never be detected as this is effectively saying that we have no idea about the systems' correct behaviour, therefore it is never possible to detect any deviation from it.

An accuracy of 95% was used, this means that small differences in the systems' and models' behaviour can be corrected by slightly modifying the models' states. If there is a larger difference in their behaviour then the models' states cannot be changed quickly enough to compensate, and the difference in the outputs will increase, causing the error index to increase and a fault will be detected.

This is what happened in the two examples where faults were detected. Small model inaccuracies were corrected by modifying the models' states but when a fault occurred it was no longer possible to modify the states by a large enough amount and hence a fault was detected.

The detection scheme does work when there is noise present and the model is not totally accurate. The balance between false alarms and sensitivity can be made by

choosing appropriate values for the accuracy of the model and the expected noise content of measurements.

7.3 Diagnosing Single Faults.

In this section two examples will be examined which demonstrate the performance of the diagnostic algorithms. Here only single faults will be looked at. The two examples looked at in the previous section will be taken, a restriction in a pipe and a leak in the tank. The diagnosis process will start at the point where a fault has been detected.

Single faults can be diagnosed by assuming that all of the components except one are functioning correctly and then trying to find a new value for the suspected faulty components' parameter. When a new value of the parameter has been found, a model of the system with this one parameter changed can be compared to the behaviour of the real system. If their behaviours are similar, then the fault (the change in the real systems' behaviour) could be explained by a change in the value of this components' parameter and therefore by a fault in this component. The new value of the parameter would indicate the nature of fault.

7.3.1 Diagnosing a restriction in the input pipe.

Here the same data as in section 7.2.2, example 1 will be used. The valve positions and flow and depth measurements are shown in figures 7.10 and 7.11. In figure 7.14, as was discussed earlier, the fault is not detected until time $\approx$ 44 minutes 30 seconds. This will therefore be the time when the diagnosis begins. Therefore, the data from time $\approx$ 44 minutes 30 seconds to time $\approx$ 82 minutes will be used.

The first part of the diagnostic process is, for each of the components, assume that it is faulty and all the other components are not faulty. A least squares estimate is then calculated for the parameter of the component. The results of this are shown

in figure 7.20. Here for the parameter for K1, it is assumed that all of the other components (except K1) are working correctly and a value is calculated for K1s' parameter. For the parameter for K7, it is assumed that all of the other components (except K7) are working correctly and a value is calculated for K7s' parameter, and so on for every component.

After this, again for each component, an error index is calculated based on a model of the system which had the new estimated value for the components parameter rather than its' original value. The error index for each component is shown in figure 7.21. The error index which is lowest will therefore correspond to the component, which when given its' new estimated parameter value will most closely match the systems' new behaviour. Remember that this is all recursive in time and that for every sample of data a new estimate for each parameter is made, and a new value for the error index is calculated.

Looking at figure 7.21 there are two which appear to approximately equally low, namely K1 and K8. In section 6.6 it was discussed how to calculate a hitting index. This is done using the error index for each hypothesis (in this case each hypothesis is for only one component). The lower the hitting index is, the better the hypothesis matches the actual system behaviour. The hitting indices calculated from the error indices in figure 7.21 are shown in figure 7.22. Here K1 and K8 are observed to indeed be the lowest of the seven hitting indices. From time $\approx$ 44 minutes 30 seconds until time $\approx$ 61 minutes, the hitting indices for K1 and K8 are virtually the same and are indistinguishable on the figure. From 61 to 66 minutes K1 is the lowest, from 66 to 77 minutes K8 is slightly lower than K1, and from 77 to 82 minutes K1 is again the lowest. From this it can be interpreted that a fault in either K1 or K8 would explain the new behaviour of the system reasonably well. Possibly K1 would be a better fit as the hitting index for K1 is substantially lower thank K8s' between 61 & 66, and

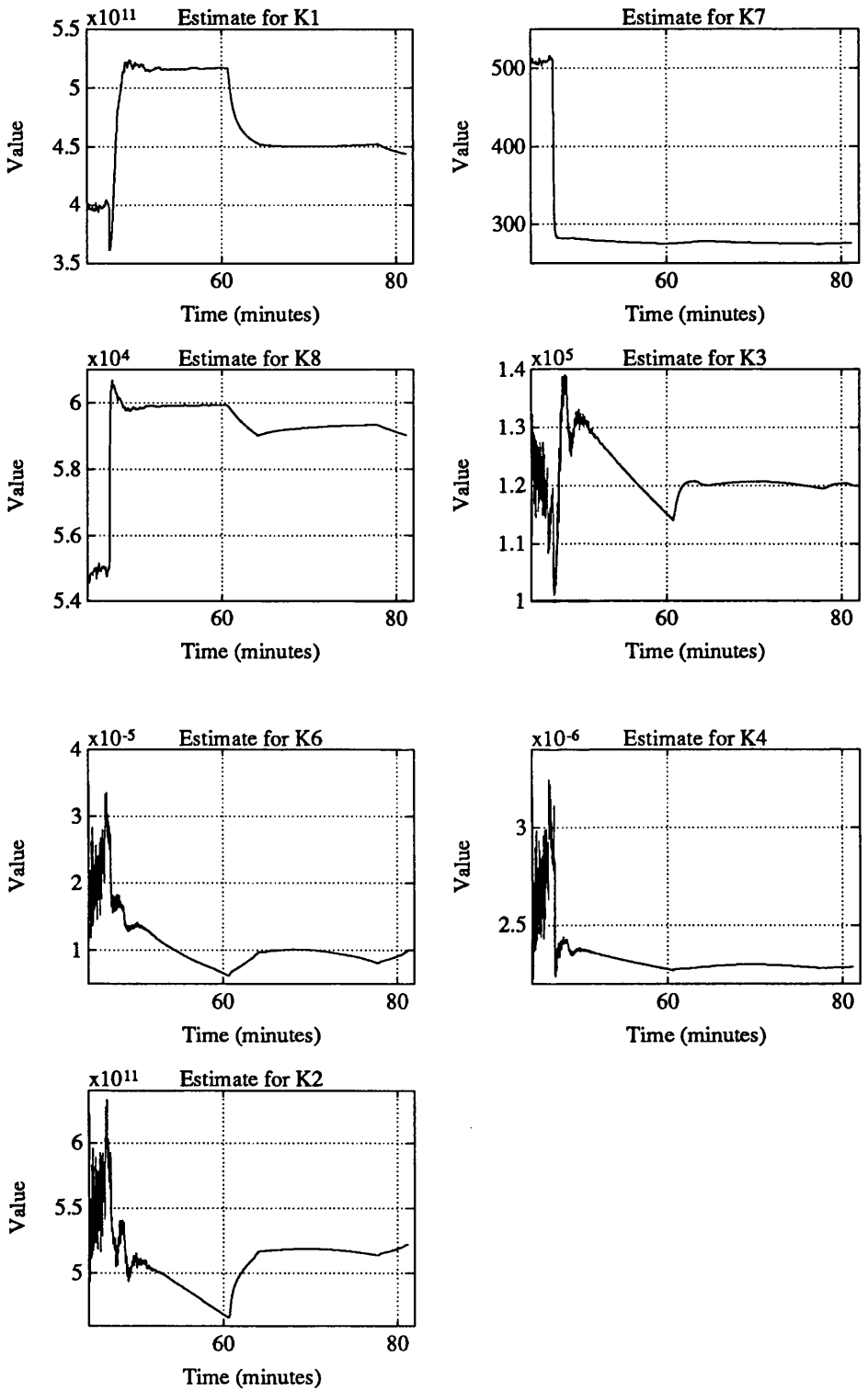


Figure 7.20: The parameter estimates for each component.

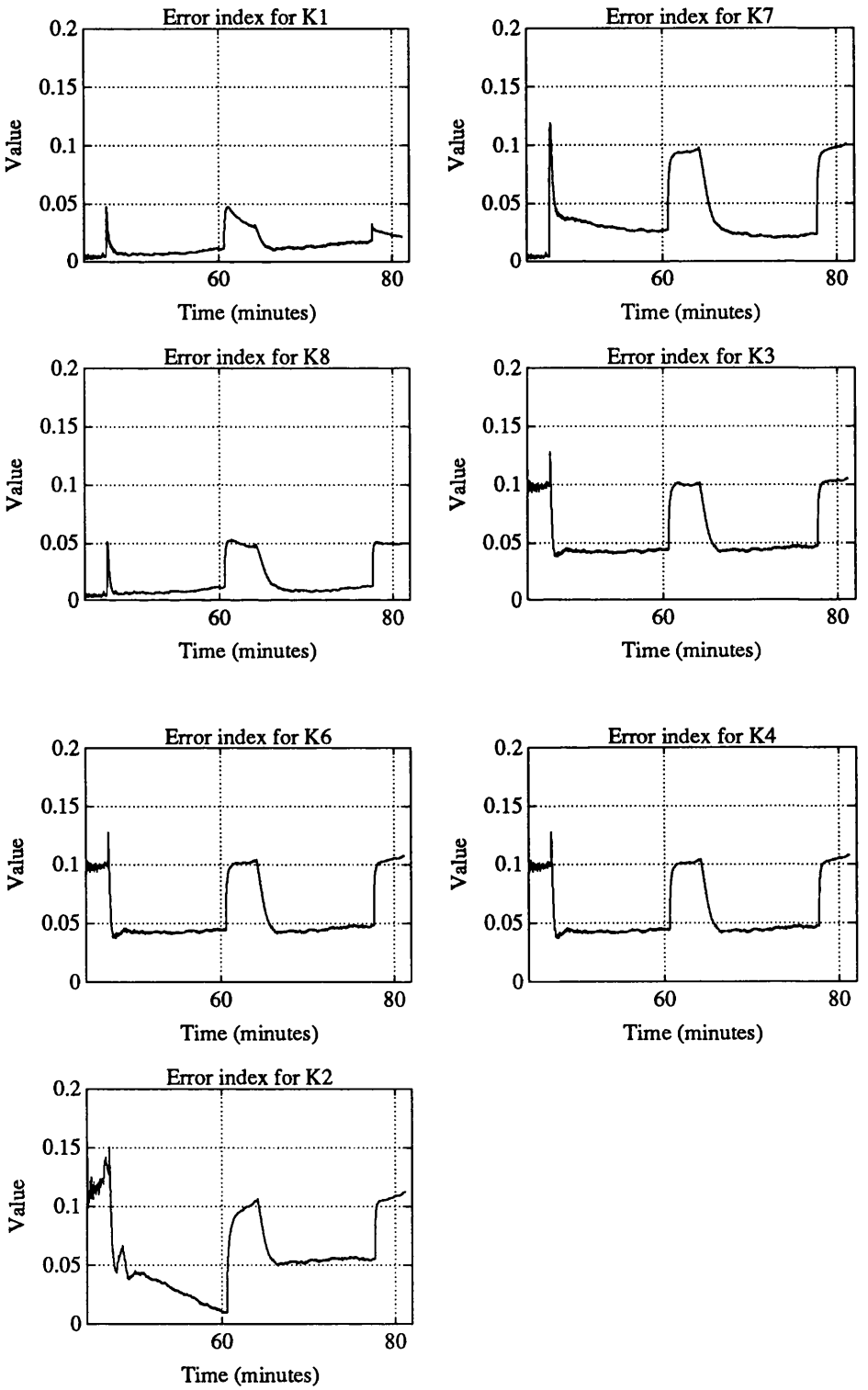


Figure 7.21: The error indices for each component.

77 & 82 minutes, whereas the hitting index for K8 is only marginly less than K1s' between 66 & 77.

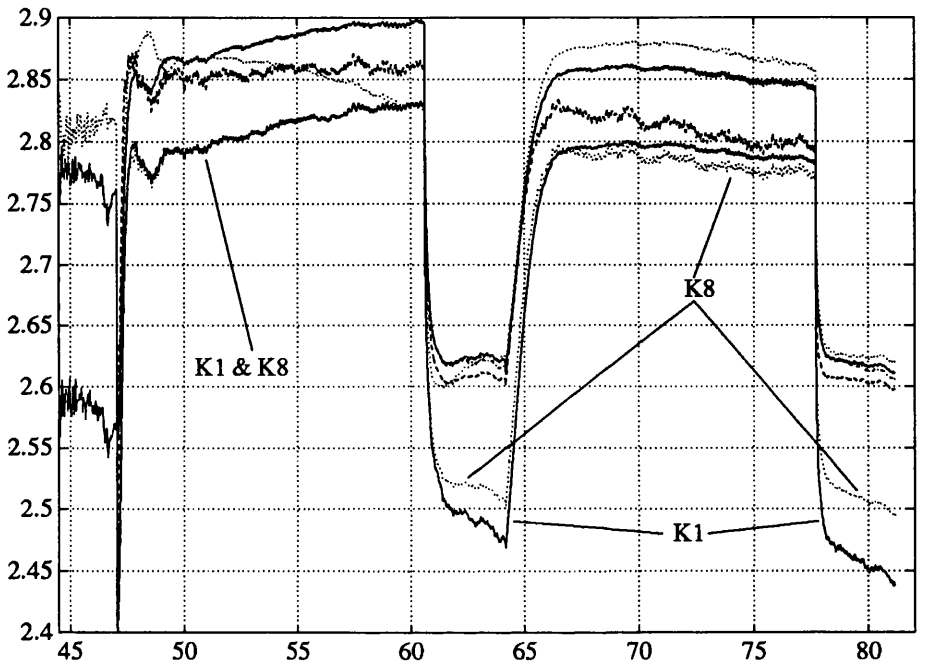


Figure 7.22: The hitting indices for each component.

So far it has been ascertained that the fault would most likely be explained by K1 being faulty, but a fault in K8 is also quite possible. Looking at figure 7.20 which shows the parameter estimates for each fault, the final estimate for K1 is 4.45×10^{11} (at time $\approx$ 81 minutes), and the final estimate for K8 is 5.5×10^{4}. Therefore, the most likely fault is a fault in K1, the pipe friction component, and the value of the parameter has increased from 2.41×10^{11} to 4.45×10^{11}, this means an increase in the pressure drop due to friction in the pipe and is analogous with a restriction in the pipe. Also possible is a fault in K8, the pressure being supplied from the pump, and the value of its' parameter has dropped from 64650 to 55000 i.e. the supply pressure has dropped causing a reduction in the flow into the tank.

To check if these are reasonable results, the same input/output data from time 41

minutes to 82 minutes is used, together with a model of the system for each of these diagnoses, with the appropriate parameter changed to the value obtained above.

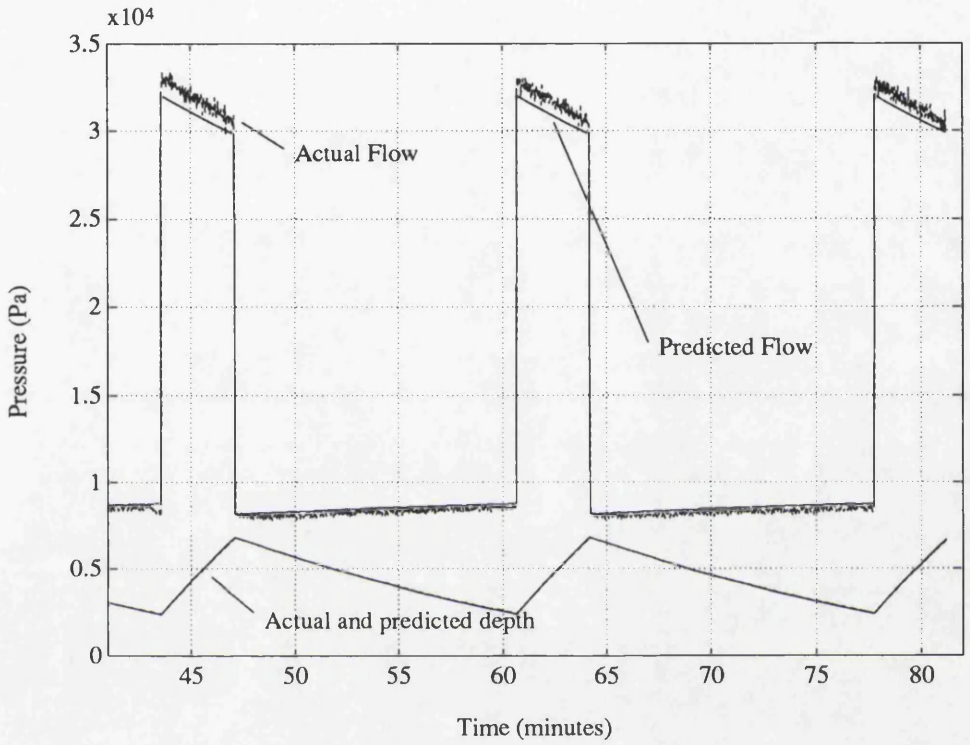


Figure 7.23: Comparing the systems behaviour with K1 faulty.

Doing this for K1 first, results in figures 7.23 and 7.24. Figure 7.23 shows how well a change in K1s' parameter to 4.45×10^{11} fits the new systems behaviour. In general it gives a similar response, the depth is almost exactly the same as the real value, and the flow is very nearly the same as the actual flow. Looking at the parameter estimate for K1 in figure 7.20, it was still changing (decreasing) at time $\approx$ 82 minutes. If the diagnosis process had been left running longer, a smaller value for K1s' parameter would have been obtained and a better fit with the actual flow in 7.23 would have been attained. If figure 7.23 is now used to calculate an error index, the one shown in figure 7.24 is obtained. The error index fluctuates around 0.02, i.e. the predicted outputs are approximately 2% in error, and remember that there is noise present which is also contributing to the value for the error index. This means that a change

in K1s' parameter from 2.41×10^{11} to 4.45×10^{11}, is a reasonable explanation the systems' new behaviour.

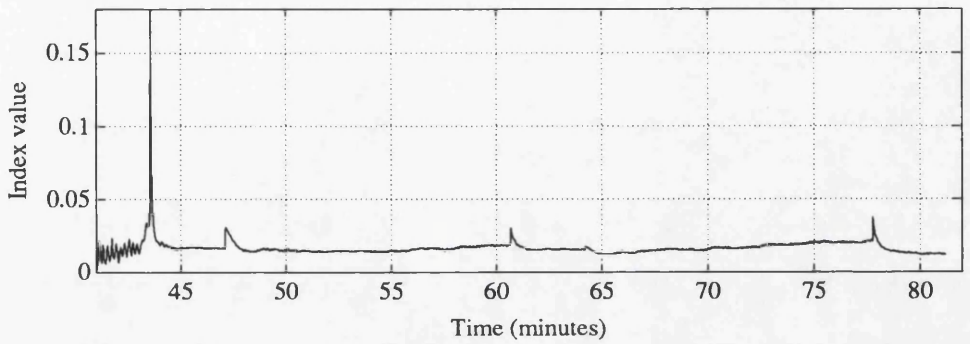


Figure 7.24: The error index for a fault in K1.

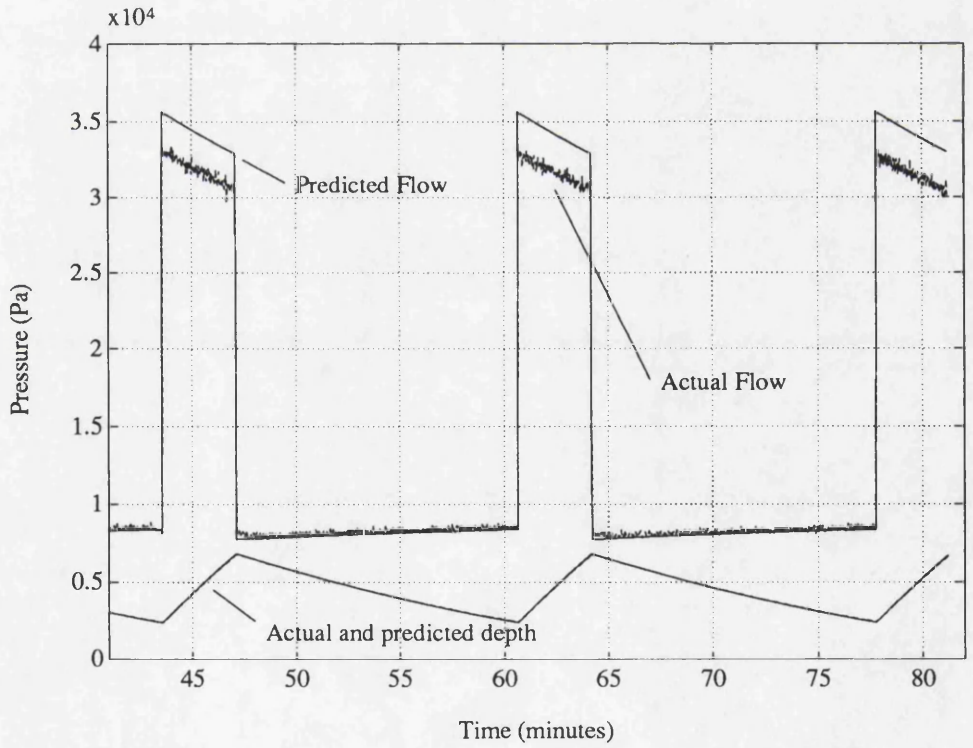


Figure 7.25: Comparing the systems behaviour with K8 faulty.

Figure 7.25 shows how well the fault hypothesis for K8, a change in the pump supply pressure, fits the actual systems' behaviour. The depth is again a very good fit, the flow is a good fit when the size of the flow is low, but at higher flows the

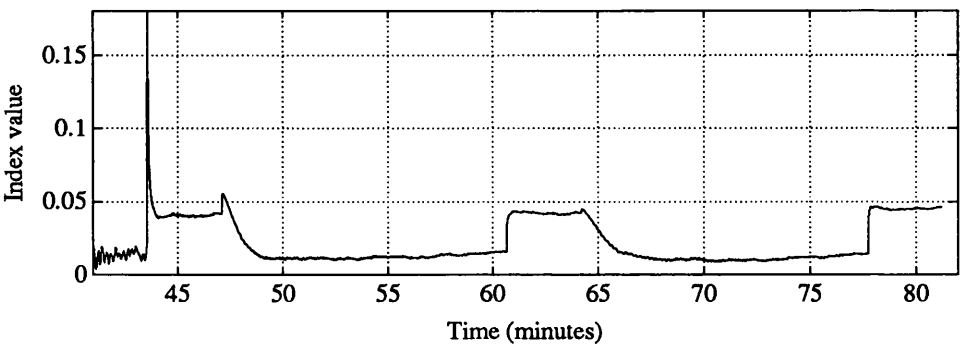


Figure 7.26: The error index for a fault in K8.

predicted flow is not as good as that given by the hypothesis K1. This is also reflected in figure 7.26 where the corresponding error index rises significantly when the flow is at a higher level.

These then confirm the results of the diagnosis that a fault in K1 (an increase in the friction coefficient) fits the behaviour of the system well, and a fault in K8 (a drop in the pump supply pressure) mostly fits but not as well as K1 does. Faults in the other components are poor fits as indicated in figure 7.21.

7.3.2 Diagnosing a leak in the tank.

In the following section the data which appeared in section 7.2.2, example 2 will be used. The input and output signals are shown in figures 7.15 and 7.16. In figure 7.19, the fault was detected at time 12 minutes and 30 seconds, so the data used here will all be from this time onwards.

In this case there is not a component which explicitly represents a leak in the tank. This could be done by inserting into the model a component with the constitutive relationship of a hole in the tank and with a parameter which corresponds to the size of the hole. Initially the value for the parameter would zero, i.e. no hole. A hole could then be indicated by an increase in this parameter. It will be noted how the diagnosis processes handles this situation.

The components' parameters are calculated in the same way as for the previous diagnosis, and they are shown in figure 7.27. The corresponding error indices are shown in figure 7.28. The lowest index is for K6, although the error index for K4 is also very low, and most of the other error indices are also low. This is because the leak from tank is quite small and although the systems' behaviour has changed, it has not changed by a great deal.

The hitting indices appear in figure 7.29. The lowest one, after time $\approx$ 16 minutes, is the index for K6, and the index just above that is the index for K4. The top index is for K2.

From this the conclusion can be drawn that the most likely explanation for the fault is that K6 has changed from 1.04×10^{-5} to 8.7×10^{-5}. This parameter represents part of the output valves behaviour, and looking back to system model, figure 7.3,page 161, this represents a constant flow out of the tank. K6 has increased from $1.04 \times 10^{-5} m^3/s$ to $8.7 \times 10^{-5} m^3/s$, this is equivalent to an additional 0.0766 litres per second flowing out of the tank, and this is therefore the size of the leak.

K4 is a constant of proportionality between the output valve position and flow out of the tank. If this hypothesis is considered, then from figure 7.28, K4 has increased from 2.35×10^{-6} to 3.65×10^{-6}, i.e. for the same valve position the flow has increased. The hitting index indicates however that K6 gives a better representation of the systems' new behaviour than K4.

To check these results, figure 7.30 show how the model with the new value for K6s' parameter compares to the real system, the error index is shown in figure 7.31. The predicted behaviour is very close to the real behaviour and the error index is low. This shows that the change in K6s' parameter is a good representation of fault.

Figure 7.32 shows the comparison between the real system and a model with K4s' parameter changed. Again the fit is good, but slight differences between the depths

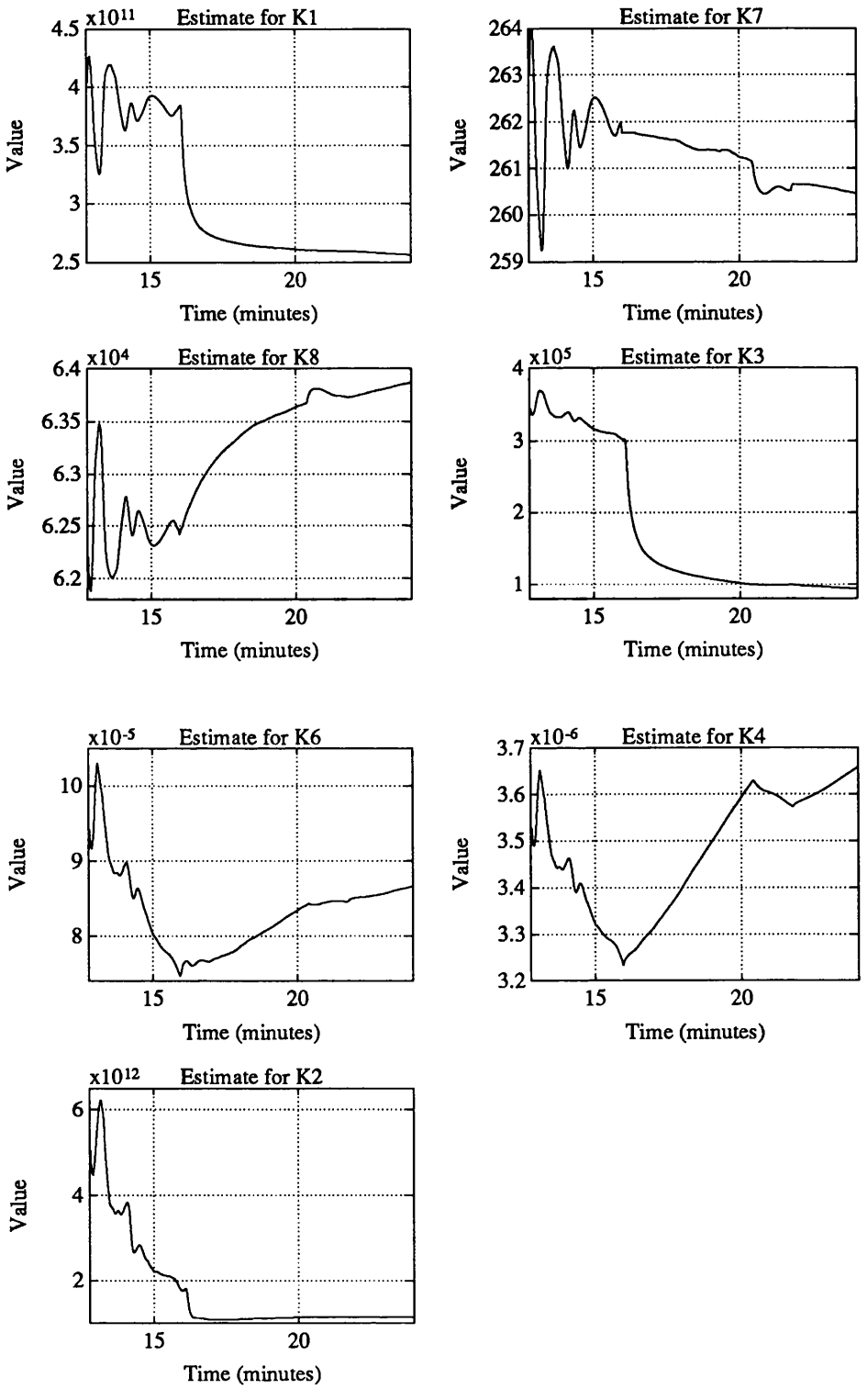


Figure 7.27: The parameter estimates for each component.

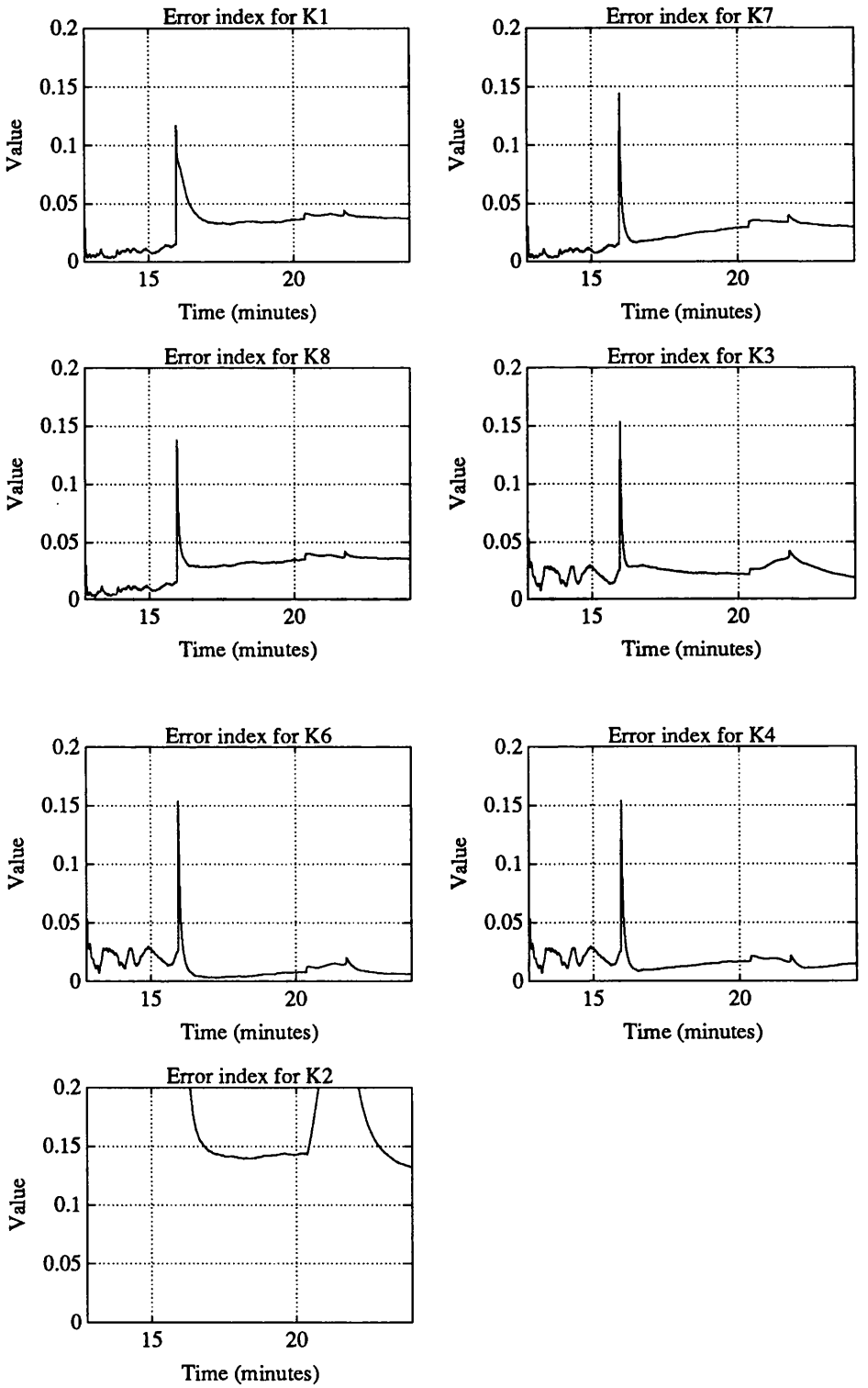


Figure 7.28: The error indices for each component.

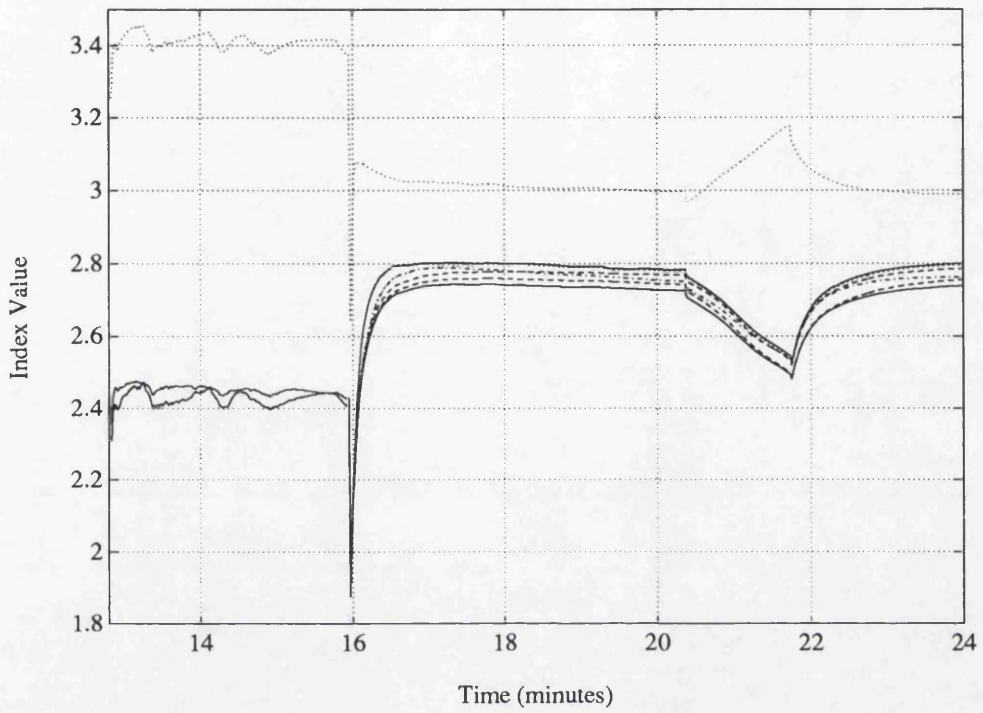


Figure 7.29: The hitting indices for each component.

can be seen after 18 minutes. The error index is shown in figure 7.33.

As stated at the beginning of this section, the model does not have a component which corresponds to a leak in the tank but, because the leak was fairly constant, the diagnostic process was able to attribute the new behaviour in an increase in K6. However, if a system was being monitored in which it was desired to accurately detect leaks, then it would be more appropriate to incorporate a suitable component into the model.

Summary

Here two examples have been shown of diagnosis with data taken from a real system. The induced faults were not very large, and there was noise present in the measurements, this meant that there were other hypotheses that also produced low hitting indices, however the lowest one in both cases did correspond to the actual

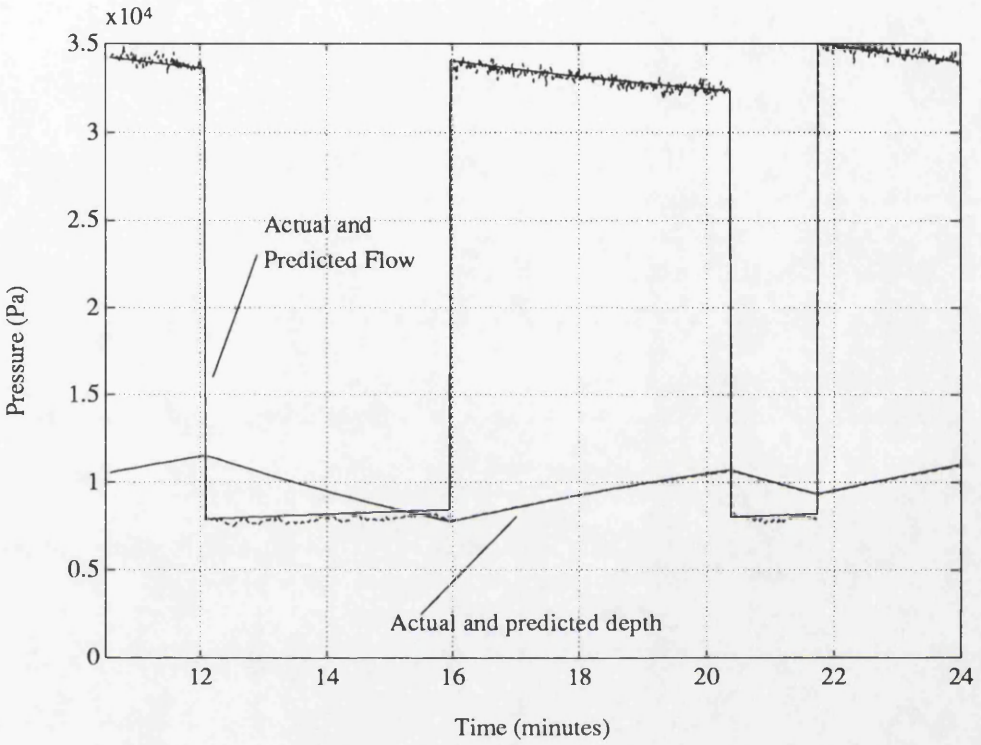


Figure 7.30: Comparing the systems behaviour with K6 faulty.

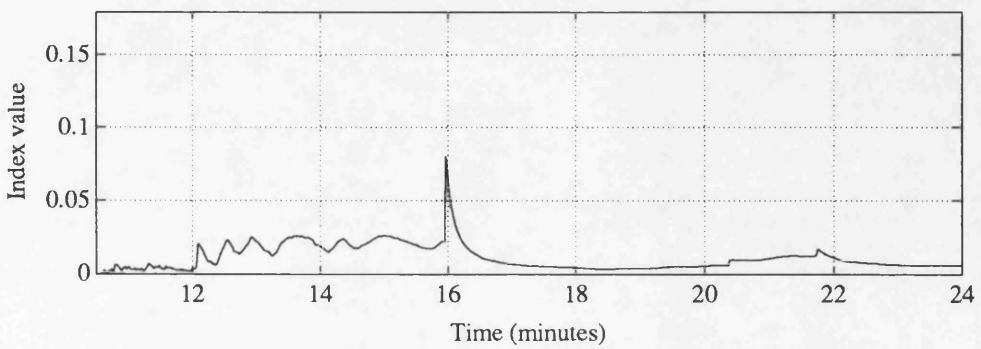


Figure 7.31: The error index for a fault in K6.

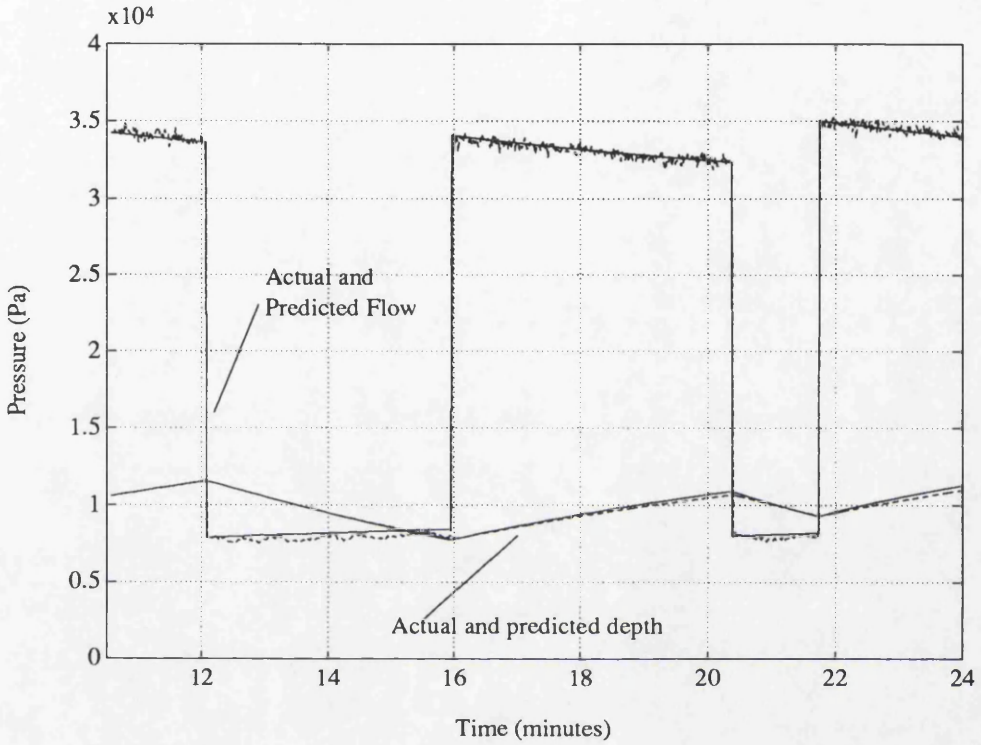


Figure 7.32: Comparing the systems behaviour with K4 faulty.

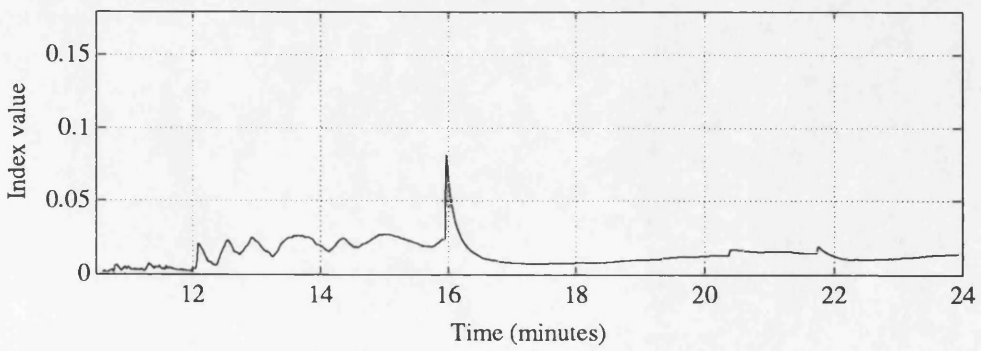


Figure 7.33: The error index for a fault in K4.

fault with the system.

The parameter estimates were shown to be reasonable by using them in a model of the system, and seeing how this model with revised parameter estimates compared to the system faulty behaviour. In each case the model behaviour was found to be close to the systems behaviour, given the approximations in the other components' constitutive relationships and parameters, and the noise in the output measurements.

7.4 Considering Multiple Faults.

In this section details of an example diagnosis are given when the diagnosis algorithm is set up to look for faults in two components simultaneously. The same data that was used earlier (when the fault was a restriction in the in-flow pipe) will be used. Here all of the hypotheses which have two components will be tested using parameter estimation and then comparing the predicted outputs and actual outputs of the system as before. However, this will not be done for single component hypotheses, but as was explained in section 6.6, the results of the two component hypotheses will be used to reason about the single component hypotheses.

In total there are 21 combinations for two component faults, there are also 7 single component faults possible. As explained earlier when trying to deal with multiple faults the number of combinations possible becomes very large, and testing each one of these is computationally intensive, so these tests should be kept to a minimum. Twenty one hypotheses will be tested here, but reasoning about 28 hypotheses will take place i.e. a saving of 7 additional tests has been made. The saving in the number of tests required, depends upon how many components are in the system, and the maximum number of simultaneous component faults that are being examined.

Figures 7.10 and 7.11 show the systems' inputs and outputs. Because of the large amount of information produced by testing 21 hypotheses, it is not practicable

to present it all here. Instead the information relevant to the hypothesis that K1 is faulty will be examined, as this was the actual fault, and it will be shown that the diagnosis process produces this as the most probable cause of the systems' new behaviour.

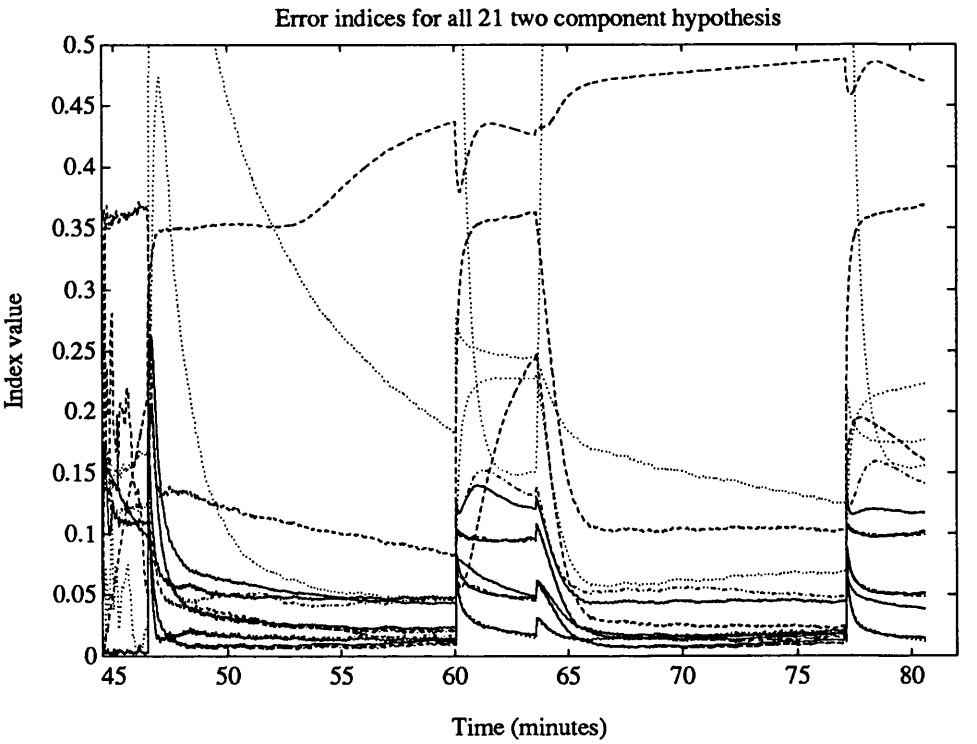


Figure 7.34: The two component error indices.

Figure 7.34 shows the error indices calculated for all of the 21, two component hypotheses. What should be noted here that there are many of these which give low values and no single diagnosis stands out as being most most probable. Figure 7.35 shows the hitting indices for the same 21 hypotheses. Again, no single hypothesis stands out as being lower than the rest.

From the hitting indices in figure 7.35 the hitting indices are computed for the single component hypotheses, as described on page 152. These are shown in figure 7.36. It can be seen clearly the hitting index for K1 is lowest, and the index for K8 is the second lowest. Also, comparing this with figure 7.35, it is seen that

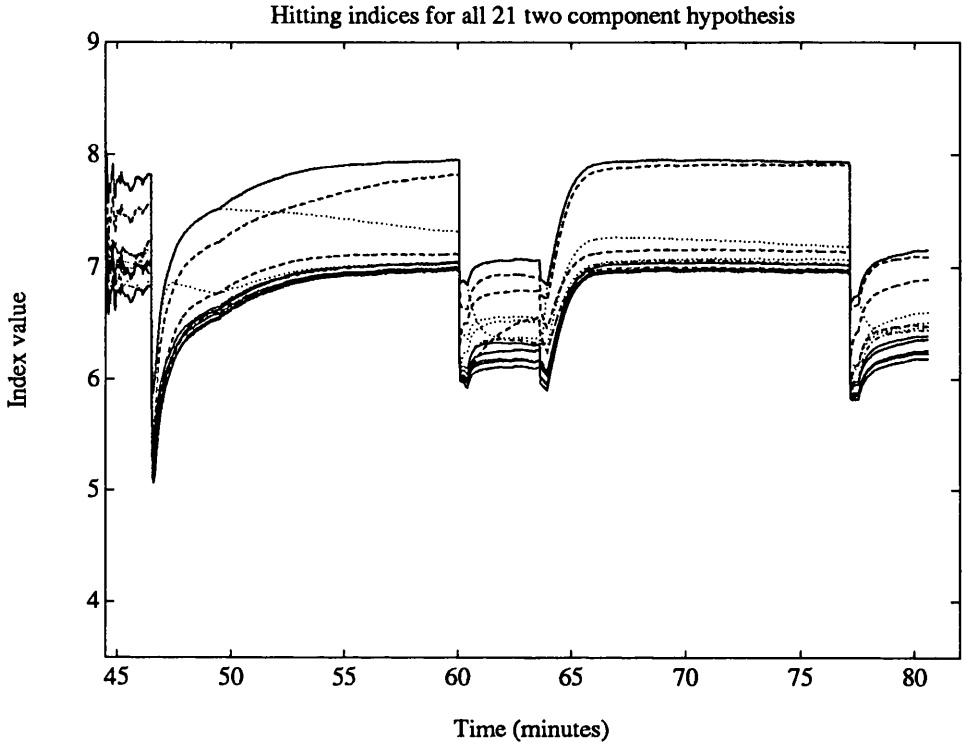


Figure 7.35: The two component hitting indices.

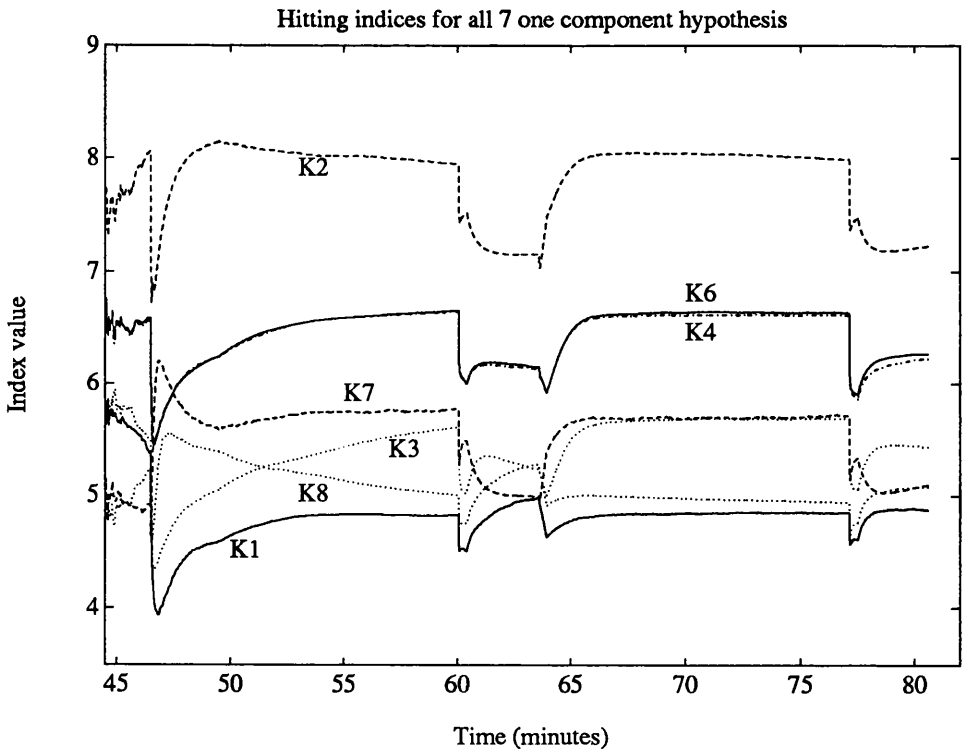


Figure 7.36: The single component hitting indices.

the hitting index for K1 is substantially lower than for any of the two component hypotheses. One can therefore be confident that a fault in K1 is the most likely explanation for the systems' behaviour. Now K1s' parameter must be found.

To find K1s' parameter all of the hypotheses tested which were supersets of K1 must be considered. Each one of these tests calculated a value for K1s' parameter. K1 will now be given a value by finding the average value for K1s' parameter, as calculated by each of these tests. The result for this is shown in figure 7.37. To show that this a reasonable estimate of K1s' parameter, this can be compared with the values obtained for K1 in figure 7.20 when only a single fault diagnosis was considered. These are seen to be very similar.

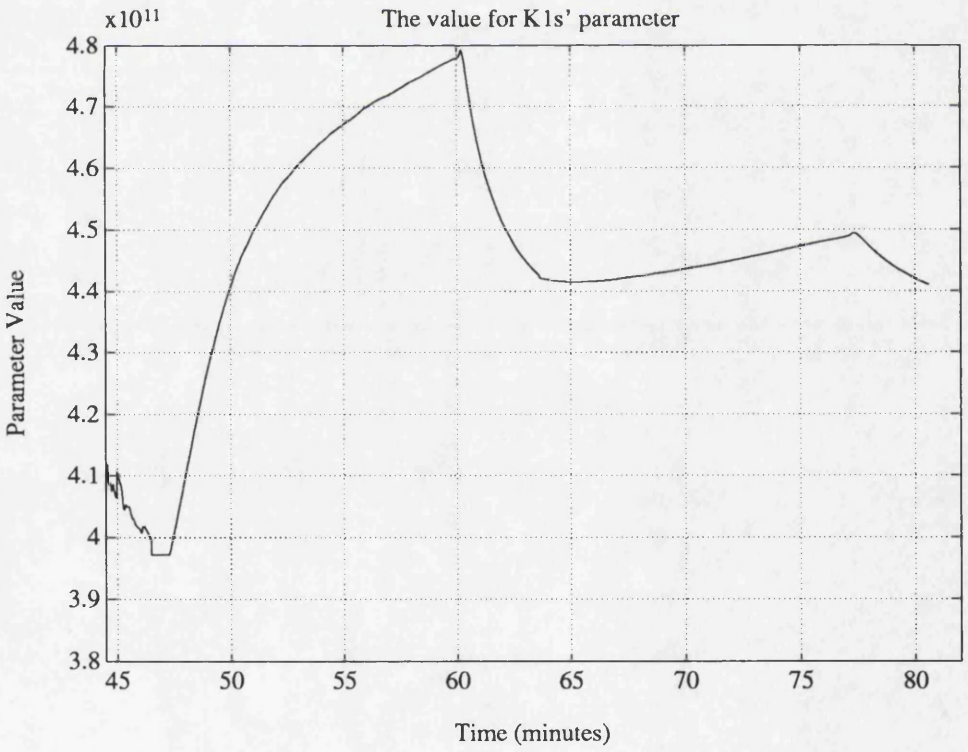


Figure 7.37: The average parameter estimate for K1.

If the fault was indeed due to two faults, one of the hitting indices for the two component hypothesis would have been substantially lower than the rest, and the hitting indices for the single component faults would have been much larger.

Summary

Here an attempt was made to show how the diagnosis process works when it considers the possibility of multiple faults. Because of the large amount of data produced, it was not possible to go into great detail here, but the diagnosis processes did correctly identify the fault and produce a good estimate for the new value for K1s' parameter.

7.5 Diagnosing Multiple Faults

In this last section, the results obtained when a multiple fault occurs are given. The faults induced in the system are as before, but this time they occur simultaneously. That is, a leak occurs in the tank and the pipe from the pump to the tank becomes partially blocked.

The measured inputs and outputs can be seen in figure 7.38. The actual outputs are the solid lines and the expected outputs are the dashed lines. The two faults occurred at approximately time = 10 minutes. At times before 10 minutes, there is little difference between the actual and expected output values. Shortly after the 10 minute mark, an error appears between the actual and expected flow. This is then followed by an error between the actual and expected water depth. The error index (in figure 7.39) crossed its limit at around 13 minutes, and a fault is detected at time 13 minutes and 12 seconds.

The diagnosis process takes place exactly as described in the previous section. Error indices for each two component hypothesis are calculated, and from these a hitting index for one and two component hypotheses are calculated. These hitting indices are shown in figure 7.40. Looking at this it can be seen that there are four hitting indices which are consistently low. These are (in order of minimum magnitude) {K1, K6} {K1, K4} {K8, K3} {K8, K6}.

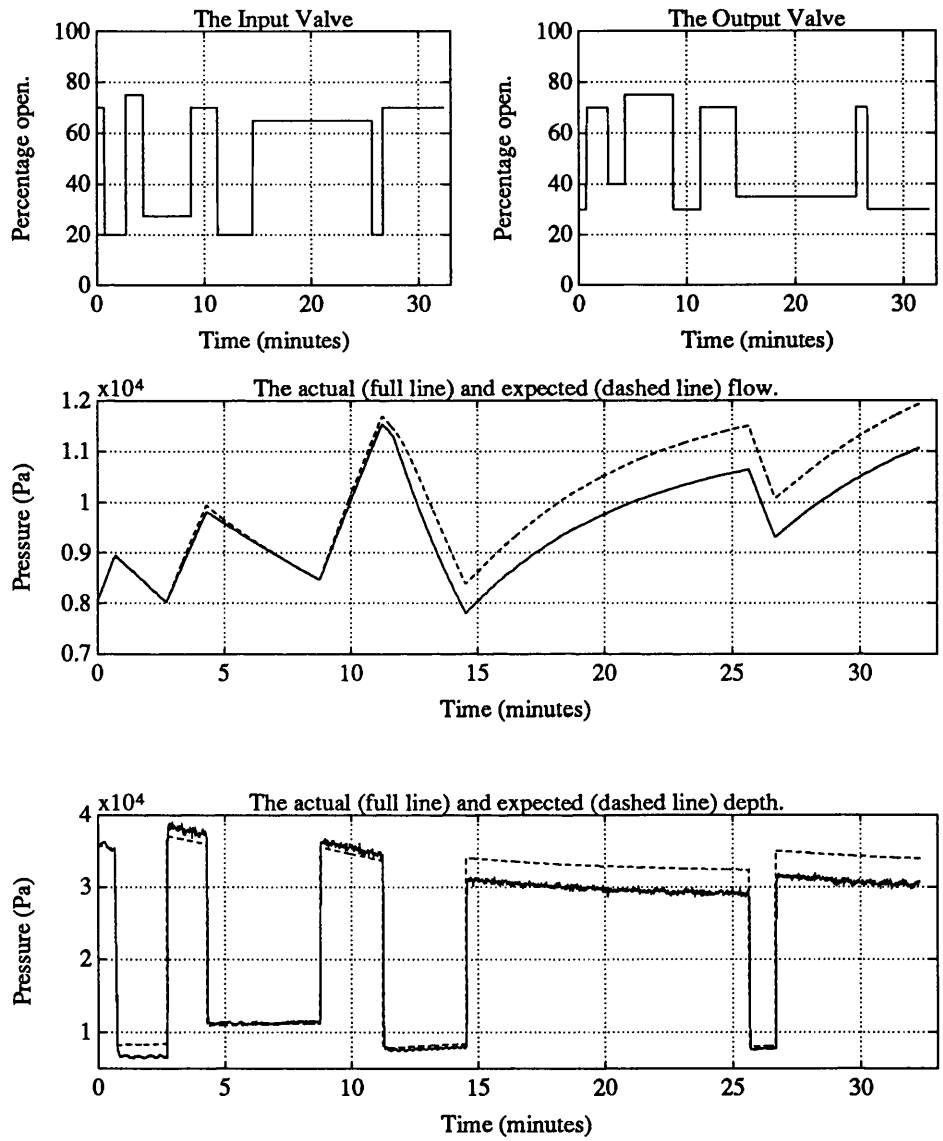


Figure 7.38: The sensor measurements and expected outputs.

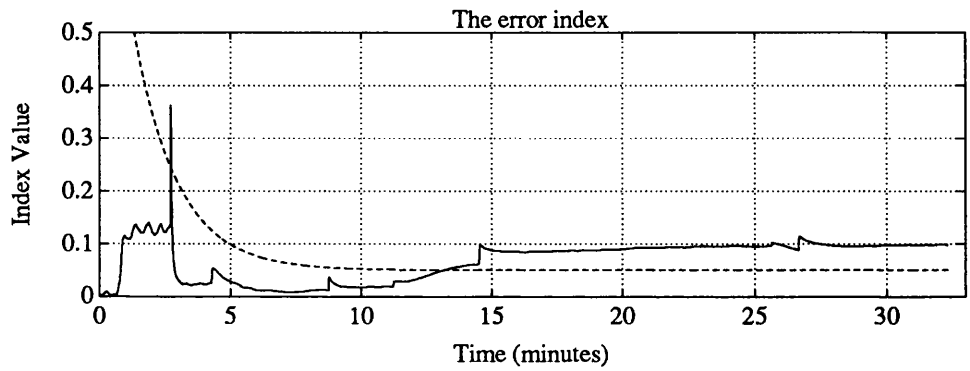


Figure 7.39: The error index and its' limit.

- {K1, K6} This has the lowest hitting index, and corresponds to an increase in the pipe friction in the pipe from the pump to the tank (K1) and an increase in the static parameter for flow through the output valve (K6). This is effectively what the genuine fault is, a blockage in the input pipe, and an increase in flow out of the tank.

- {K1, K4} This has the second lowest hitting index, and corresponds to an increase in the pipe friction in the pipe from the pump to the tank as before (K1), and an increase in the proportional parameter for flow through the output valve (K4). This is also effectively what the genuine fault is.

- {K8, K3} This has the third lowest hitting index. This is indicated as being likely, because a fault in these components has a similar effect as the effect of the genuine faults. Firstly K8 (the pump supply pressure) is indicated as being a candidate because if K8 were to decrease, then the result would be a smaller flow into the tank. As it happens the real fault was due to a restriction in the input pipe, but this also has the effect of reducing the flow into the tank. Secondly K3 (the tanks parameter) is indicated as being a candidate because if K3 were to increase, this would correspond to an increase in the tanks cross-sectional area. The result of that would be that the level in the tank would not rise as quickly for the same in flow. The actual fault was a leak in the tank, which has the same effect, namely that the level of water in the tank does not rise as quickly as expected.

- {K8, K6} This has the fourth lowest hitting index. K8 is indicated for the reasons just described, and K6 is indicated because, as also described above, it represents a constant out flow from the tank, and the behaviour of the leak is similar to this.

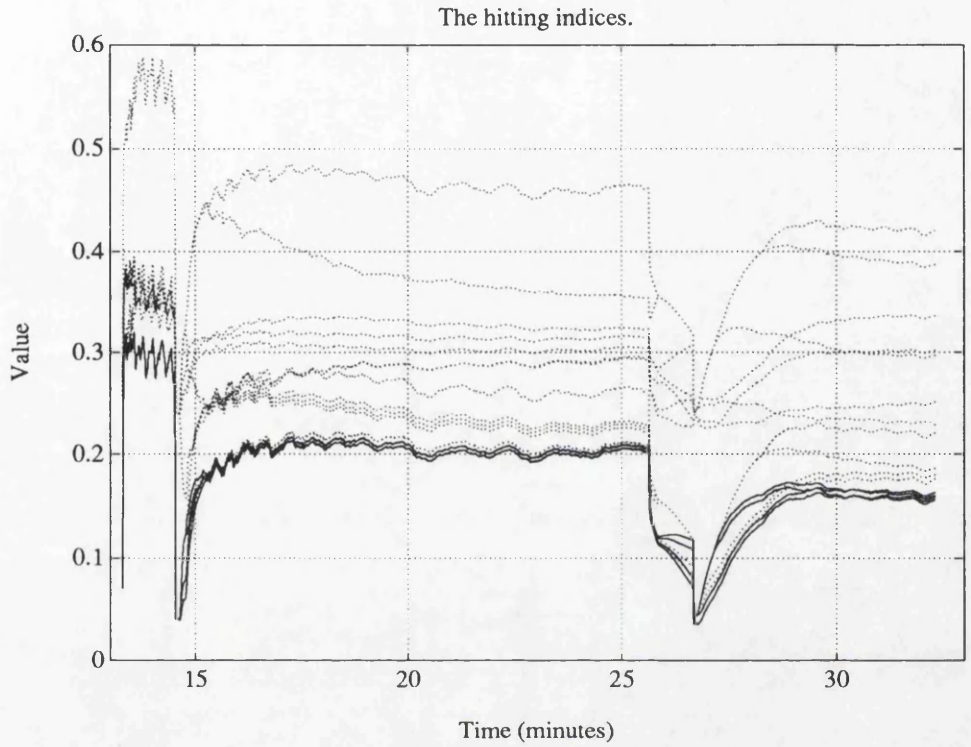


Figure 7.40: The hitting indices for all one and two component hypotheses.

Summary

To summarise these results, the lowest two hitting indices correspond well to the actual faults that were induced on the system. Other hypotheses which also had low values would produce similar symptoms to the actual faults. As the size of the faults were not excessive, and the system was not accurately known, it was not easy to differentiate between these, although the lowest hitting index did accurately represent the faults.

Chapter 8

Conclusions.

Fault detection and fault diagnosis in dynamic engineering systems have been examined. Constraint propagation techniques [52] [51] [27], have been used together with established model based approaches to fault detection [41] [12] [21] and fault diagnosis [62] [19] [74] [36] [8] [70]. In addition aspects of multiple fault diagnosis theory [68] [10] [11] have been drawn upon to enable us to reason about multiple faults and to do this in a way which does not vastly increase the computing overheads, although an increase is required.

Although the overall detection and diagnosis system is similar to that of others, an attempt has been made to cope with model inaccuracies, noisy measurements and systems where the states of the system are not necessarily measurable. The result of this is a recursive in time diagnosis algorithm which can be used on-line rather than a post-process fault diagnosis system. The faults which this system was designed to diagnose must be representable as a change in a components' parameter, rather than a new, undefined behaviour. Therefore, to show that one or more components are faulty, it is necessary to evaluate their new parameters and show that the new values are consistent with the systems' behaviour. This has been advantageous for two reasons, since it has allowed the exploitation of the dynamic nature of the systems being

considered and this has enabled the identification of single and multiple component faults in systems where the number of sensors are limited. Previous implementation of constraint propagation methods have not been able to do this.

Our implementation consists of an off-line system analysis section, and an on-line fault detection and diagnosis section. The off-line section extracts system equations from the model and generates a self contained C++ source code which contains all of the algorithms necessary to detect and diagnose faults in a particular system. This can be ported to any machine with C++. This has allowed us to shift the bulk of the computation to the once only off-line part of the system. This has resulted in a practicable implementation which can be used on-line, and in real time.

Through simulations and tests on a real system, it has been shown that the approach adopted here is capable of detecting and diagnosing faults. In a noisy, imprecisely known system it is impossible to be 100% sure that a diagnosis is correct. The system therefore orders the hypotheses, so that the most likely ones can be checked first.

In chapter 2 a survey of other fault detection and fault diagnosis methods was presented. The origins of constraint propagation techniques and different approaches to system modelling were also given.

In chapter 3 the background to the bond graph modelling techniques were given. These enabled the models of systems to be made much more quickly and easily. Bond graphs can be used to represent systems from more than one domain, in a standard format, enabling the fault analysis algorithms to work on mechanical, electrical, hydraulic or thermodynamic systems, or any system which combines any of these domains.

Also shown was how the constraint propagation methods have been used in fault detection and diagnosis, and some of the limitations with their use in these ways were

highlighted. Notably their inability to cope with noisy systems, inaccurate system models and dynamic system where not all of the systems' states are measurable. Dynamic components were briefly introduced, and it was shown that by using the measurements from more than one time interval, the rates of change could be resolved. The combinatorial problem of considering multiple fault hypotheses was also discussed and it was shown that for a system with N components, there were $2^N - 1$ possible component combinations.

Chapter 4 described how to propagate signals through time and the model, and also described the propagation algorithm. This algorithm, when given some known signals, can find the shortest route from any unknown signal to these known signals, in such a way that it is possible to solve all of the constitutive relationships of the components' which are en route, to yield the unknown signals or parameters in terms of the known signals and known parameters. An overview of the software which has been developed was given. This indicated the different roles of the off-line and on-line sections of the software. Also described in detail was the filtering method used on the measured signals, before input to the rest of the detection/diagnosis system. Using this filtering method greatly increased the performance, both in terms of speed and accuracy, of the parameter estimation and therefore the diagnosis algorithm. It also reduced the likelihood of a false alarm being triggered by the presence of noise.

In chapter 5 the method for detecting faults in dynamic systems was described. This consisted of predicting what the systems' outputs should be, given the system model and the measured system inputs. The predicted system outputs were then compared to the actual system outputs and an error index was calculated which reflected, on average, the percentage difference between the two sets of outputs. If this error index exceeded a limit determined by the measurement noise content, a fault was indicated. While this was happening, the models' states were being adjusted

according to the actual system outputs, and how accurate our model of the system was. In this way modelling inaccuracies could be compensated for by adjusting the states to reduce the difference between the predicted and actual outputs.

In chapter 6 the fault diagnosis method was discussed, and it was shown how to determine how well an estimated parameter, in a component suspected to be faulty, reflected the systems' behaviour. This formed the basis for a dynamic system theorem prover which is compatible with that described by Reiter [68]. This enabled us to go on and consider multiple fault diagnoses. In addition to this a method was proposed for reducing the significant amount of computation required by only testing a subset of all of the multiple faults to be checked, and then extrapolating this information to reason about hypotheses which were not explicitly tested. Some of the limitations of this were outlined when the methods were applied to real systems, this was chiefly that system equations needed to be solvable. Finally determination of which hypotheses were the mostly likely was explained, as it is not possible to be absolutely certain about any of the hypothesis.

In chapter 7 all of these methods were put to the test on a real system. The system consisted of a water tank, a pump and an assortment of pipes, control valves and presure sensors. The system was noisy and approximations were made in modelling the system. Firstly the detection algorithm was tested on the system when no fault was present. Happily, no fault was detected. This showed that the modelling errors and noisy measurements did not produce a false alarm.

The detection algorithm was then tested on two examples of a fault. In each case, shortly after the fault occurred, a fault was detected. These faults were small changes in the system, namely a slight restriction imposed in a pipe and a small leak in the tank.

Next the diagnosis algorithm was shown finding which component failure most

accurately described the systems faulty behaviour, and the estimates of the parameter associated with the fault were shown. This was done for both of the faults described above.

The extrapolation algorithm performance was shown when considering the possibility that multiple faults could occur. 21 two component hypotheses were tested, and then the decision that the fault was caused by a single component failure was taken, and an estimate of its new parameter value was produced. This was even though single component fault hypotheses were not explicitly being tested. This showed that the number of computationally expensive hypotheses tests could be reduced, without reducing the number of hypotheses which could be reasoned about. This makes the diagnosis of multiple faults possible in real time, and helps reduce the combinatorial problem when multiple faults are considered.

Finally, the diagnosis of two separate faults occurring at the same time was investigated. Hitting indices for single and double faults were calculated, and the lowest hitting indices were those from multiple fault hypotheses which would produce similar symptoms to those which the faulty system was exhibiting. The hypothesis with the lowest hitting index was also the one which most closely represented what the actual faults were.

Although the algorithms work, there is no doubt that their performance could be increased by a more sophisticated parameter estimation method and an improved hitting index calculation algorithm.

Appendix A

Automatic Constraint Generation.

Here the workings of the automatic constraint generation algorithm will be discussed in detail. Information which must be kept during the propagation is identified and two alternatives for the strategy for finding the shortest possible route for propagating are discussed. One of these is substantially faster than the other, but also could require considerably more memory to implement successfully. A detailed example will then be shown of how the constraints for a particular situation are automatically generated. The DC motor will be used again, and the constraints will be generated which will allow the calculation of the value for the resistors' parameter (R) and the value for the inductance of the motors' windings (L), assuming that both of these are unknown, and the applied voltage and the speed of the motor are being measured.

A.1 Information requirements during propagation.

During the automatic constraint generation, an account must be kept of information which describes the current state of the propagation process. This was implemented in prolog, and the information was represented in lists, the algorithm will therefore

be explained with relation to lists. The propagation which is described here starts at any unknown signals or parameters that are to be found, and then a route is found to known measurements or parameters. When such a route has been found, propagation takes place backwards along this path, from the known variables, to the unknowns to be evaluated. Here the first stage of this is looked at which involves propagating from the unknowns to the knowns.

At all times during propagation, the following information must be stored.

1. A list containing which parameters or signals to be found. This changes throughout the propagation process, at the beginning it is a list containing the unknowns of primary interest which are to be found. As propagation proceeds, the list contains what must be found in order to be able to calculate the initial signals and parameters which were in the list. At the end of propagation, the list will be empty, indicating that enough information has been found to calculate the original objectives. This list will be called U, a list of **unknowns** to be found.

2. A list which contains the parameters and signals which are known. This list will be called K, the **known** parameters and signals.

3. A list which contains the components which may be propagated through. Any one component may be used only once to propagate through, so whenever one is used, it must be removed from this list, so that it cannot be used again. This list will be called C, the list of **components** which may be used for propagation.

The need and use of these will be described further over the following sections.

A.2 Finding the shortest possible path.

Before we begin with the actual propagation itself, lets first look at an important area which must be covered by the implementation. The path that is found from unknown values to known parameters or signals must be the shortest possible path. If not the path will unnecessarily involve other components making the final solution more complex and making it more time consuming to solve during the on-line detection and diagnosis process. This problem arises because, during propagation, there will quite frequently arise situations in which there is a choice of which component to propagate through next. i.e. from the list U, K and C, the situation will be that the signal x is to be found, and this could be done by propagating through component A or through component B. At this stage in the propagation there is no way of telling which, if either, is the better path to take.

A.2.1 Our method for finding the shortest path.

The solution adopted here to overcome this problem is, whenever there is a choice of two or more paths to take, all of these path are taken, independently, but simultaneously. To explain what this means examine figure A.1. $P = 0$ is the initial position where no propagation has taken place. U, K and C will be set up indicating what must be found, what is known and which components are to be used. These are indicated by U_1, K_1 and C_1. The algorithm/program then goes through a one component propagation step to $P = 1$, where one component has been propagated through, and there was no choice to make about which component to propagate through. List U, K and C will have change to reflect the change in the current state of propagation to U_2, K_2 and C_2. Another one component propagation step can now be made. This time there is a choice, and there are two alternatives which can be taken, taking both of these leads to the state where there are two sets of values for

U, K and C, $\{U_3, K_3, C_3\}$ and $\{U_4, K_4, C_4\}$. Both of these are equally valid without further information, and therefore both of them must be explored. For the next one component propagation step, both of these sets will be used separately, firstly looking at $\{U_3, K_3, C_3\}$, and propagating through one component giving $\{U_5, K_5, C_5\}$ and then through $\{U_4, K_4, C_4\}$ giving $\{U_6, K_6, C_6\}$. It is of course possible that either or both of these could have resulted in another choice giving even more sets of U, K and C which need to be investigated. This process continues as shown in figure A.1, with the number of single component propagations to be done at each stage gradually increasing.

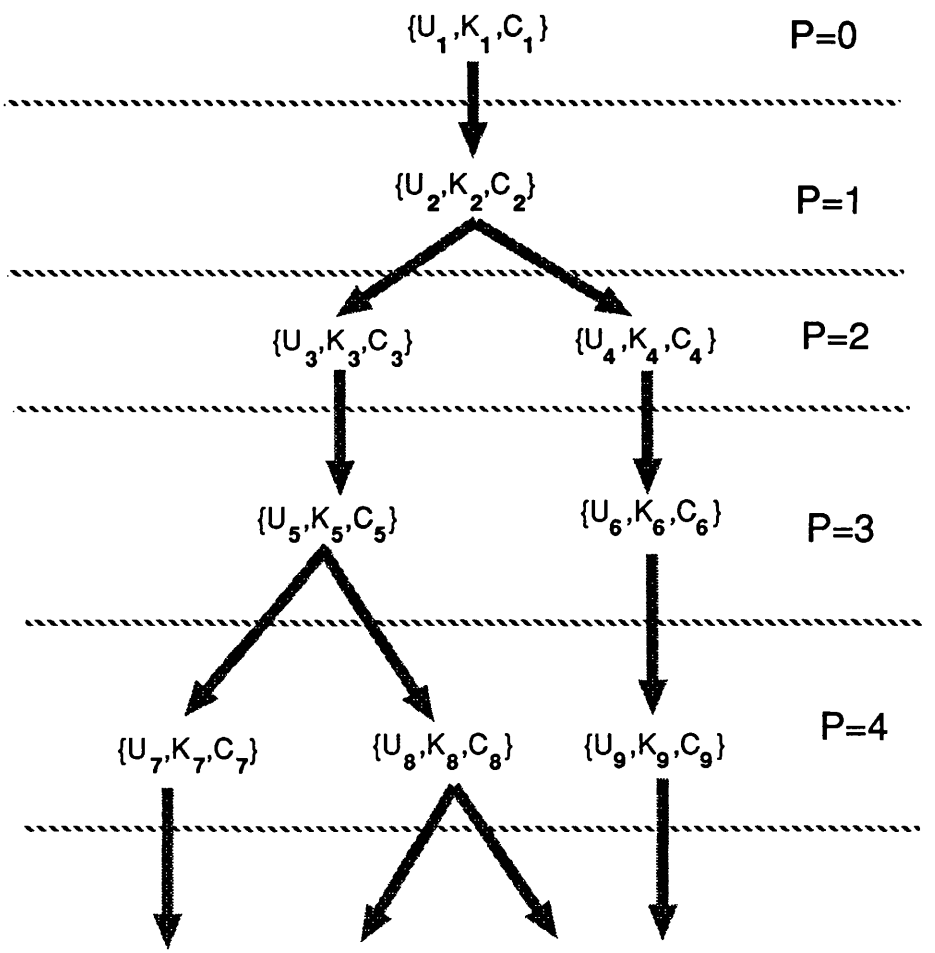


Figure A.1: The propagation tree.

At first it would seem that the number for different propagation routes is enormous

and that this rapidly increasing number of routes to be checked would quickly consume large amounts of memory for storing all of these sets of U, K and C, and the program would slow down drastically as the amount of processing for each step increased. Although it is true that the number of alternative routes does increase, it is possible to 'prune' this 'tree' of possible routes as it is being created thereby reducing the actual number of routes which need to be checked. The first point to note is that different routes may in fact be considered as being the same. For example if propagating from known measurements to find a signal, one route may go along the route $A \Rightarrow B \Rightarrow C \Rightarrow D \Rightarrow E$, whereas another route might be $C \Rightarrow D \Rightarrow B \Rightarrow A \Rightarrow E$. These are both different routes, but they involve going through exactly the same components, so later when all of these five components' relationships are solved for the signal to be found, exactly the same answer is obtained for each of the routes. i.e. going from known data, through the same component to a signal. If the known data is the same for each route, and each route goes through the same components, only in a different order, then the result for the signal, when taken together, will be the same for each route. One of these routes can therefore be safely deleted without a the final outcome. This also means that any additional choices of routes originating from the one which is deleted, will never be created.

Additionally, a restriction that propagation may only take place backwards through time is imposed, and a limit is put upon how far backwards in time to propagate. Therefore if a route goes backwards in time too far, it will be deleted, also if a route goes backwards in time from the current time, e.g. propagates through a state, and later tries to propagate forwards in time through another state, it is prevented from doing so. This reduces how often valid choices are available. If the only route possible involved the violation of one of these restrictions, then obviously that whole route is deleted since it has no path which it is allowed to take. If the

list, U (the unknowns), is not empty, but the list, C (the component which may be propagated through), is empty, then this means some unknowns are still to be found, but there are no more components to propagate through[1]. This route is therefore deleted as it has failed to find a valid path.

The whole of the above process continues until, for any route, the list U becomes empty, that is, there are no more unknowns to be found for that path. Since every possible valid route had been taken from the original unknown parameter or signal, the one for which U becomes empty first must also be the shortest route.

It is possible for all of the routes to be deleted before any of them have successfully managed to reduce their unknown list, U, to the empty list. In this case there is no valid route and a solution for the unknown cannot be found. This would happen, for example in the DC motor if only the applied voltage is measured, and the value of R was required given these voltage measurements, then there is no way to find a solution. Various routes will be explored, but eventually they will all be deleted as being invalid. This is why a limit on how far backwards in time the algorithm is allowed to go is imposed. Otherwise, in this situation, the algorithm would go on forever going further backwards in time looking for another known value so that it would be able to solve the equations.

A.2.2 An alternative method.

As an alternative to trying to travel down all of the possible routes simultaneously, one route could be taken and backtracking employed every time the route turned out to be an incorrect one. In this case the first valid route found could not be guaranteed to be the correct one. A note would have to be made of the route and the backtracking employed again to find all of the possible valid routes and then pick the

[1]There is one exception to this in the implementation which will be discussed later.

smallest one from these. Doing this would probably reduce the memory requirements for the algorithm, but it would certainly greatly increase the time for the shortest route to be found, as every possible route would have to be taken to prove whether or not it was a valid one. The algorithm use here stops as soon as the first valid route is found, this also must be the shortest path. When this happens there could be many other routes which are still under investigation in memory and the will be others which have not been started yet, but as the shortest, valid path has been found, all of these others can be discarded.

A.3 An example of propagation.

To help show how the automatic constraint generation algorithm works, figure A.2 will be referred to, this is a block diagram of a DC motor showing three consecutive periods of times as before, but all of the components and all of the connections between the components have been indicated. The connections between components are labelled 'a' through 'm', these will be referred to as signals. Each state in figure A.2 is shown as consisting of three components, namely the *parameter*, the change in time between the time frames dt, and a summing junction which sums the previous state with the change in the state to give the current state. In the model this is represented as two components, the *parameter* being one, and the change in time & summing junction being the other. The shaded areas labelled Sa and Sb represent these. The two other summing junctions have been given the labels Aa and Ab. When referring to a particular component or connection, the time frame will be appended to the label of the component or connection. e.g. if to refer to the inertia (J) in the current time frame J_0 would be used, for the previous time frame J_1 is used, etc. therefore Ab_2 would refer to the summing junction Ab in the time frame labelled 'Time = 2'.

In the initial situation the lists U, K and C will have the following values.

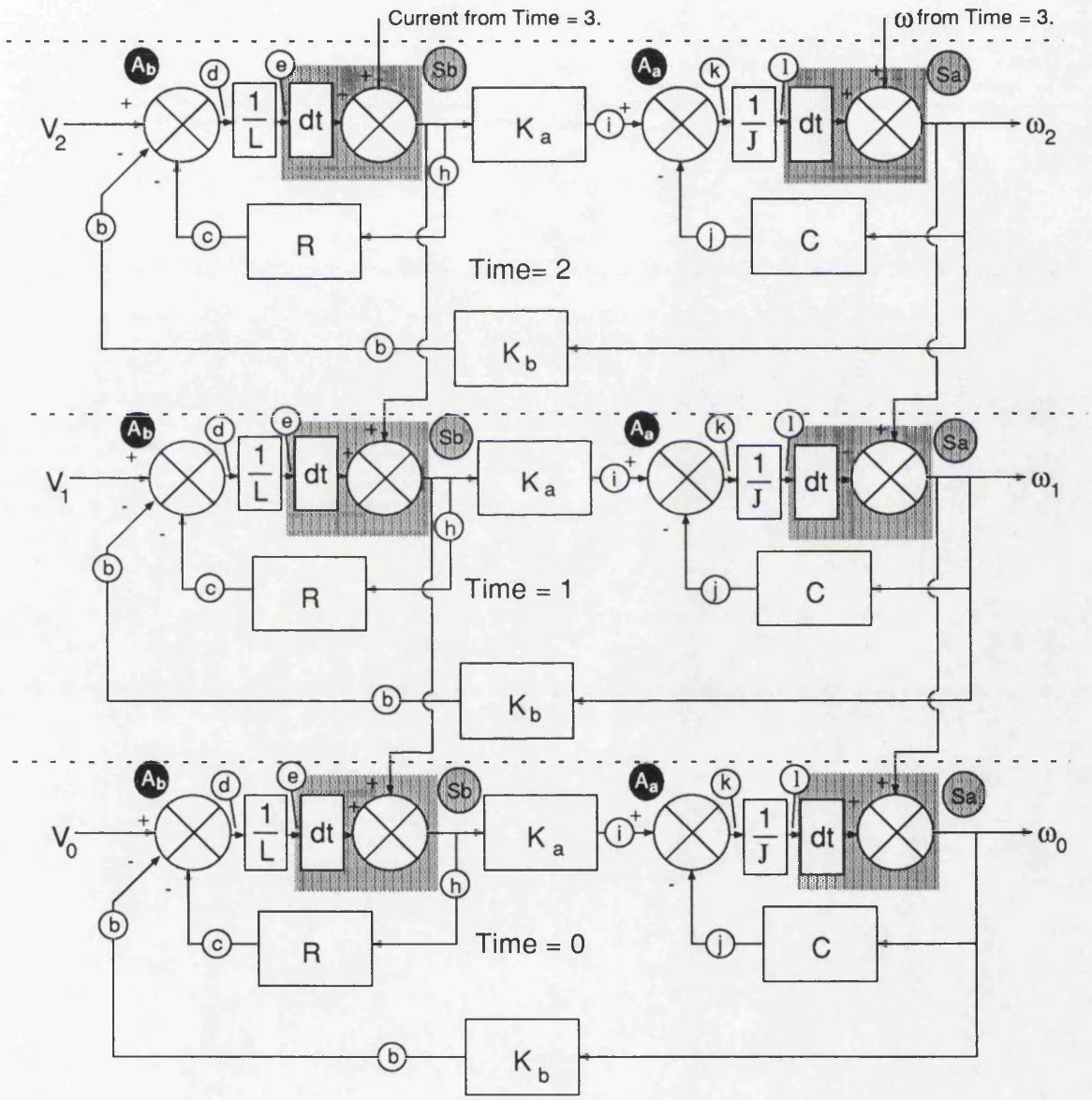


Figure A.2: A block diagram of a DC motor.

- $U = \{R, L\}$, the unknowns.
- $K = \{V_x, \omega_x, Ka, Kb, J, C\}$, the known parameters and measurements. V_x indicates that V is know for any value of x i.e. all the values for V for every time period are known. To indicate that the current value of V was known, but no other values for V were known, then the list would contain V_0 instead of V_x. Parameters need no subscript as a parameter is constant for every time step.
- $C = \{L_0, R_0, Ka_0, Kb_0, J_0, C_0, Aa_0, Ab_0, Sa_0, Sb_0\}$. These are not to be confused with parameter values. Here, L_0 means that the relation for L for the current time step has not been used for propagation, it is independent of whether the parameter L is known or unknown.

The algorithm begins by first looking at the first element of U, this is the parameter R, to find R (the parameter) the relationship of R (the component) must be used. From figure A.2 it is seen that in order to find R (the parameter), the values of c_0 and h_0 need to be known. The three list U, K and C can now be updated.

1. $U = \{c_0, h_0, L\}$.
   $K = \{R, V_x, \omega_x, Ka, Kb, J, C\}$.
   $C = \{L_0, Ka_0, Kb_0, J_0, C_0, Aa_0, Ab_0, Sa_0, Sb_0\}$.

You will notice that R in U has been replaced by c_0 and h_0, since these are needed to find R. R has appeared in K, since R would be known if the values the members of U were known. R_0 has been removed from C since its relationship was used in finding that R was dependent on c_0 and h_0. After each step the list U is sorted according to the following rules.

1. Any duplicates in U are removed.
2. If U contains any unknown parameters, they are all placed at the end of the list. The order of the parameters themselves is not important. Since the algorithm

always starts at the beginning of U, this means that the algorithm will only attempt to find a parameter once all of the signals which are unknown have been found. This results in the algorithm following a path from the unknowns to be found, to the known values. When that path is complete, the next parameter in U is used and this process is repeated.

3. The signals are sorted according to which time period they refer to. e.g. e_0 refers to the signal e in the present time period and e_1 refers to the signal e in the previous time period. e_0 will come before e_1. This means that an earlier time period will only be used if there is no other route through the current time period.

The above step is then repeated. Looking at c_0 which is the first member of U. Since R_0 has been removed from C there is no choice but to use Ab_0 to find c_0. Doing this produces the following results.

2. $U = \{V_0, b_0, d_0, h_0, L\}$
   $K = \{c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
   $C = \{L_0, Ka_0, Kb_0, J_0, C_0, Aa_0, Sa_0, Sb_0\}$.

The process is then repeated, looking at the first member of U again. This time V_0 is the first member of U, its value is known since it is measured and V_x appears in K. It can then be removed from U to give $U = \{b_0, d_0, h_0, L\}$. The values of U, K and C the change as follows.

Propagating through Kb_0.

3. $U = \{Kb, \omega_0, d_0, h_0, L\}$
   $K = \{b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
   $C = \{L_0, Ka_0, J_0, C_0, Aa_0, Sa_0, Sb_0\}$.

In U, Kb is a known parameter, and so it can be removed.

4. $U = \{\omega_0, d_0, h_0, L\}$.
   $K = \{b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
   $C = \{L_0, Ka_0, J_0, C_0, Aa_0, Sa_0, Sb_0\}$.

In U, ω_0 is a measured signal, it is therefore known and it can be removed from U.

5. $U = \{d_0, h_0, L\}$.
   $K = \{b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
   $C = \{L_0, Ka_0, J_0, C_0, Aa_0, Sa_0, Sb_0\}$.

Propagating through L_0.

6. $U = \{L, e_0, h_0, L\}$.
   $K = \{d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
   $C = \{Ka_0, J_0, C_0, Aa_0, Sa_0, Sb_0\}$.

In U, L now appears twice. One of these occurrences of L can therefore be removed.

After sorting, U will be as shown below.

7. $U = \{e_0, h_0, L\}$.
   $K = \{d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
   $C = \{Ka_0, J_0, C_0, Aa_0, Sa_0, Sb_0\}$.

Propagating through Sb_0.

8. $U = \{h_1, h_0, h_0, L\}$.
   $K = \{e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
   $C = \{Ka_0, J_0, C_0, Aa_0, Sa_0\}$.

One of the duplicates in U, h_0, is removed.

9. $U = \{h_1, h_0, L\}$.
   $K = \{e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
   $C = \{Ka_0, J_0, C_0, Aa_0, Sa_0\}$.

U is sorted putting h_0 before h_1.

10. $U = \{h_0, h_1, L\}$.
    $K = \{e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
    $C = \{Ka_0, J_0, C_0, Aa_0, Sa_0\}$.

Propagating through Ka_0.

11. $U = \{Ka, i_0, h_1, L\}$.
    $K = \{h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
    $C = \{J_0, C_0, Aa_0, Sa_0\}$.

Ka is removed from U as this parameter is known.

12. $U = \{i_0, h_1, L\}$.
    $K = \{h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
    $C = \{J_0, C_0, Aa_0, Sa_0\}$.

Propagating through Aa_0.

13. $U = \{k_0, j_0, h_1, L\}$.
$K = \{i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{J_0, C_0, Sa_0\}$.

Propagating through J_0.

14. $U = \{J, l_0, j_0, h_1, L\}$.
$K = \{k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{C_0, Sa_0\}$.

J is removed from U as this parameter is known.

15. $U = \{l_0, j_0, h_1, L\}$.
$K = \{k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{C_0, Sa_0\}$.

Propagating through Sa_0.

16. $U = \{\omega_0, \omega_1, j_0, h_1, L\}$.
$K = \{l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{C_0\}$.

Both ω_0 and ω_1 are measured values and so they are both removed from U.

17. $U = \{j_0, h_1, L\}$.
$K = \{l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{C_0\}$.

Propagating through C_0.

18. $U = \{C, \omega_0, h_1, L\}$.
$K = \{j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{\}$.

C is known and ω_0 is known, they are both removed from U.

19. $U = \{h_1, L\}$.
$K = \{j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{\}$.

At this point C is empty and U still contains unknowns. Above it was stated that in this situation propagation ceases, there is however one situation in which propagation does not cease. That is when, after U has been sorted, the first element of U refers to a time period that has not been descended to yet. In this case the first element of U is h_1. When this happens, all of the components from this new time period are inserted into C. So C becomes as shown below. This is done because at the beginning of propagation it is not known how far backwards in time it will be necessary to go. It would be possible at the beginning to put all of the components from the last 20 or even 40 time periods into C. This just means that large amounts of data will be manipulated throughout the propagation, although it is possible to do this, it was considered undesirable from a program execution point of view.

20. $U = \{h_1, L\}$.
$K = \{j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{L_1, R_1, Ka_1, Kb_1, J_1, C_1, Aa_1, Ab_1, Sa_1, Sb_1\}$

To find a value for h_1 there is the choice to propagate through Ka_1, R_1 or Sb_1. As stated earlier in these situations all three of these routes are taken. Each one of these would give U, K and C as follows :-
Propagating through Ka_1.

21. (a) $U = \{Ka, i_1, L\}$.
$K = \{h_1, j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{L_1, R_1, Kb_1, J_1, C_1, Aa_1, Ab_1, Sa_1, Sb_1\}$.

or

Propagating through R_1.

(b) $U = \{R, c_1, L\}$.
$K = \{h_1, j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{L_1, Ka_1, Kb_1, J_1, C_1, Aa_1, Ab_1, Sa_1, Sb_1\}$

or

Propagating through Sb_1.

(c) $U = \{e_1,h_2,L\}$.
$K = \{h_1, j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{L_1, R_1, Ka_1, Kb_1, J_1, C_1, Aa_1, Ab_1, Sa_1\}$.

Each of these sets of $\{U, K, C\}$ would now be followed simultaneously, here though, only results of propagating though Ka_1 will be followed.
So, following on from 21(a) above removing Ka and propagating through Aa_1.

22. $U = \{k_1, j_1, L\}$
$K = \{i_1, h_1, j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{L_1, R_1, Kb_1, J_1, C_1, Ab_1, Sb_1, Sa_1\}$.

Propagating through Sa_1.

23. $U = \{\omega_1, \omega_2, j_1, L\}$
$K = \{k_1, j_1, h_1, j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{L_1, R_1, Kb_1, J_1, C_1, Ab_1, Sb_1\}$.

Now removing ω_1 and ω_2 from U and propagating through C.

24. $U = \{\omega_1, L\}$
$K = \{j_1, k_1, j_1, h_1, j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{L_1, R_1, Kb_1, J_1, Ab_1, Sb_1\}$.

Removing ω_1 from U gives.

25. $U = \{L\}$
$K = \{j_1, k_1, j_1, h_1, j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$.
$C = \{L_1, R_1, Kb_1, J_1, Ab_1, Sb_1\}$.

Now U contains only L. Looking at C it can be seen that L_1 has not been used yet. If it had been used, all of the components from the next time period would be added to C e.g. L_2, R_2, etc. So now, starting at L_1 it is necessary to find the values of d_1 and e_1.

26. $U = \{d_1, e_1\}$
$K = \{L, j_1, k_1, j_1, h_1, j_0, l_0, k_0, i_0, h_0, e_0, d_0, b_0, c_0, R, V_x, \omega_x, Ka, Kb, J, C\}$
$C = \{R_1, Kb_1, J_1, Ab_1, Sb_1\}$.

It should now be clear how this will continue, from d_1 and e_1 propagation continues until C is empty. Since all of the possible routes are taken, the first one in which C becomes empty is the smallest. If a record is kept of all the components passed through and all of the variables which were to be found, then this information can be used to generate a mathematical description of how R and L are functions of ω_0, ω_1, ω_2, ω_3, V_0 and V_1. This would look something like table A.1.

All of these can then be decomposed to the actual mathematical relationships (e.g. $c_0 = R \times h_0$) and then all of these equations must be solved, simultaneously if necessary. To do this, all of these equations are printed out into in a file which is

Unknown	Found by using component	Found by using known values
R	R_0	c_0, h_0
c_0	Ab_0	V_0, b_0, d_0
b_0	Kb_0	Kb, ω_0
d_0	L_0	L, e_0
:	:	:
:	:	:
l_2	Sa_2	ω_2, ω_3
j_2	C_2	ω_2, C

Table A.1: All of the components needed to solve in order of propagation.

then fed into symbolic equation solving package. This solves all of the equations and produces results, which for the example used here, would be of the form $L = f_1(\omega_0, \omega_1, \omega_2, \omega_3, V_0, V_1)$ and $R = f_2(\omega_0, \omega_1, \omega_2, \omega_3, V_0, V_1)$.

In this case,

$L = \frac{A}{B}$

$R = \frac{C}{D}$

where

$$
\begin{aligned}
A &= dt^2K(Cdt\omega_1V_0 - CdtV_1\omega_0 + \omega_2JK\omega_0 - \omega_2JV_0 - JK\omega_1^2 + J\omega_1V_1 + J\omega_1V_0 \\
&\quad - JV_1\omega_0) \\
B &= C^2\omega_2dt^2\omega_0 - C^2dt^2\omega_1^2 + C\omega_2dtJ\omega_1 + 2C\omega_2dtJ\omega_0 - C\omega_3dtJ\omega_0 - 2CdtJ\omega_1^2 \\
&\quad - \omega_2^2J^2 + \omega_2J^2\omega_1 + \omega_2J^2\omega_0 + \omega_3J^2\omega_1 - \omega_3J^2\omega_0 - J^2\omega_1^2 \\
C &= -dtK(C\omega_2dtK\omega_0 - C\omega_2dtV_0 - CdtK\omega_1^2 + Cdt\omega_1V_1 + Cdt\omega_1V_0 - CdtV_1\omega_0 \\
&\quad + \omega_2JK\omega_1 + 2\omega_2JK\omega_0 - \omega_2JV_1 - 2\omega_2JV_0 - \omega_3JK\omega_0 + \omega_3JV_0 - 2JK\omega_1^2 \\
&\quad + 2J\omega_1V_1 + J\omega_1V_0 - JV_1\omega_0) \\
D &= C^2\omega_2dt^2\omega_0 - C^2dt^2\omega_1^2 + C\omega_2dtJ\omega_1 + 2C\omega_2dtJ\omega_0 - C\omega_3dtJ\omega_0 - 2CdtJ\omega_1^2 \\
&\quad - \omega_2^2J^2 + \omega_2J^2\omega_1 + \omega_2J^2\omega_0 + \omega_3J^2\omega_1 - \omega_3J^2\omega_0 - J^2\omega_1^2
\end{aligned}
$$

These fairly complex equations which give R and L in terms of known parameters and measurements were all generated automatically by the propagation algorithm, which found the route from the known to unknown values and then passed that information to the symbolic equation solver which produced the solved equations shown above.

So, to summarise, the software is given a description of a dynamic system. It is informed which signals are measured and which are to be found. It will then find how these are related in the simplest terms (i.e. via the shortest route) and solve the equations to give the desired variables in terms of the known variables and measurements.

Bibliography

[1] A. Abu-Hanna, R. Benjamins, and Wouter Jansweijer. Device understanding and modeling for diagnosis. *IEEE Expert*, April 1991.

[2] M. Basseville and A. Benveniste. Signal processing techniques for damage monitoring in vibration mechanics. *Proceedings of the First European control Conference*, pages 2031–2036, 1991.

[3] W. Bibel. Constraint satisfaction from a deductive viewpoint. *Artificial Intelligence*, 35:401–413, 1988.

[4] D. G. Bobrow. Special volume on qualitative reasoning about physical systems. *Artificial Intelligence*, 24 (1-3), 1984.

[5] J. A. Bradshaw and R. M. Young. Evaluating design using knowledge of purpose and knowledge of structure. *IEEE Expert*, April 1991.

[6] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32:281–331, 1987.

[7] R. Davis. Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence*, 24:347–410, 1984.

[8] R. Davis, H. Shrobe, W. Hamscher, K. Wieckert, M. Shirley, and S. Polit. Diagnosis based on description of structure and function. In *Proc. of the National Conference on Artificial Intelligence*, page 137, 1982.

[9] J. de Kleer. *Local Methods for Localising Faults in electronic circuits*, volume AI memo 394. Massachusetts Institute of Technology, 1976.

[10] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32, 1987.

[11] J. de Kleer and B. C. Williams. Reasoning about multiple faults. *Science. Qualitative Reasoning and Diagnosis: Automated Reasoning*, 1987. Year may be 1986.

[12] D. Dvorak and B. Kuipers. Model-based monitoring of dynamic systems. *IJCAI*, 1989.

[13] A. Emami-Naeini, M. M. Akhter, and S. M. Rock. Effect of model uncertainty on failure detection: The threshold selector. *IEEE Transactions on Automatic Control*, 33(12):1106–1115, 1988.

[14] A. Emami-Naeini, S. M. Rock, and M. M. Akhter. Robust failure detection with varying refenence and failure times. In *Proceedings of the 27th Conference on Decision and Control, Austin, Texas*, 1986.

[15] P. K. Fink and J. C. Lusth. Expert system and diagnostic expertise in the mechanical and electrical domains. *IEEE Transactions and Systems, Man and Cybernetics*, 17(3), 1987.

[16] K. D. Forbus. Qualitative reasoning about physical processes. In *Proceedings 7th International Joint Conference on Artificial Intelligence*, 1981.

[17] K. D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[18] K. D. Forbus. Interpreting observations of physical systems. *IEEE Transactions on Systems, Man and Cybernetics*, 17(3), June 1987.

[19] P. M. Frank. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - a survey and some new results. *Automatica*, 26(3), 1990.

[20] D. W. Franke. Deriving and using decriptions of purpose. *IEEE Expert*, April 1991.

[21] M. Gallanti, L. Gilardoni, G. Guida, and A. Stefanini. Exploiting physical and design knowledge in the diagnosis of complex industrial systems. In *Proceedings 7th European Conference on Artificial Intelligence*, pages 335–349, 1986.

[22] P. J. Gawthrop. Bond graph anaysis using prolog and reduce. Control Engineering Report 89.7, Glasgow University, Dept. of Mechanical Engineering, 1989.

[23] P. J. Gawthrop. Modelling process systems with bond graphs. Control Group Research Report R-90/17, Glasgow University, Dept. of Mechanical Engineering, 1990.

[24] P. J. Gawthrop. MTT: Model transformation tools. a bond graph toolbox. Control Group Research Report R-90/13, Glasgow University, Dept. of Mechanical Engineering, 1990.

[25] P. J. Gawthrop, R. W. Jones, and S. A. MacKenzie. Identification of partially-known systems. Control Group Research Report R-90/15, Glasgow University, Dept. of Mechanical Engineering, 1990.

[26] P. J Gawthrop, R. W. Jones, and S. A. MacKenzie. Identification of partially-known systems. In *9th IFAC/IFORS symposium on identification and system parameter estimation. Budapest, Hungary*, pages 1347–1352, 1991.

[27] P. J. Gawthrop and N. A. Marrison. Fault detection, location and identification in dynamic systems. In *Proceedings of the European Control Conference: ECS91*, 1991.

[28] P. J. Gawthrop, N. A. Marrison, and L. Smith. MTT: A bond graph toolbox. In *Proceedings of the 5th IFAC/IMACS Symposium on Computer-aided Design of Control Systems:CADCS91*, 1991.

[29] P. J. Gawthrop, H. Mirab, and X. Li. Robot model validation. In *Inst. Measurement and Control 2nd Symposium on Model Validation*, London, May 1989.

[30] P. J. Gawthrop and M. T. Nihtila. Identification of time-delays using a polynomial identification method. *Systems and Control Letters*, 5:267–271, 1985.

[31] P. J. Gawthrop, M. T. Nihtila, and A. Besharati-Rad. Recursive parameter estimation of continuous-time systems with unknown time delay. *C-TAT*, 15(3), 1989.

[32] P. J. Gawthrop and L. Smith. An environment for industrial process design using bond graph modelling. In *Proceedings of the 13th IMACS World Congress, Dublin*, 1991.

[33] P. J. Gawthrop and L. Smith. *Metamodelling.* Prentice-Hall, (In preparation).

[34] P. J. Gawthrop and L. Smith. Causal augmentation of bond graphs. *Journal of the Franklin Institute*, (to appear).

[35] M. R. Genesereth. The use of hierarchical design models in the automated diagnosis of computer systems. *Stanford University Heuristic Programming Project*, HPP-81-20, December 1981.

[36] M. R. Genesereth. Diagnosis using hierarchical design models. In *Proc. of the National Conference on Artificial Intelligence*, page 278, 1982.

[37] J. Gertler and K. C. Anderson. An evidential reasoning extension of model-based failure diagnosis. *IEEE TH0282-4/89/0000/0520$01.00*, 1989.

[38] H. W. Gusgen and J. Hertzberg. Some fundamental properties of local constraint propagation. *Artificial Intelligence*, 36:237–247, 1988.

[39] N. Heintze, S. Michaylov, and P. Stuckey. Clp(r) and some electrical engineering problems. In *Fourth IEEE Symposium on logic programming - San Francisco*, 1987.

[40] S. H. Hoh, P. Thorpe, K. Johnson, and K. F. Martin. Sensor based machine tool condition monitoring system. Technical report, University of Wales Institute of Science and Technology, P O Box 23, Cardiff Cf1 3XE, 1988.

[41] R. Isermann. Process fault detection based on modelling and estimation methods. a survey. *Automatica*, 20:387, 1984.

[42] J. Jaffar and J. L. Lassez. Constraint logic programming. In *Fourth IEEE Symposium on logic programming - San Francisco*, 1987.

[43] J. Jaffar and S. Michaylov. Methodology and implementation of a clp system. In *Fourth IEEE Symposium on logic programming - San Francisco*, 1987.

[44] D. C. Karnopp. Bond graphs in control: Physical state variables and observers. *J. Franklin Institute*, 308(3):221–234, 1979.

[45] D. C. Karnopp and R. C. Rosenberg. *System Dynamics: A Unified Approach.* John Wiley, 1975.

[46] A. M. Keuneke. Device representation (the significance of functional knowledge). *IEEE Expert*, April 1991.

[47] Jee Hong Kim and Zeungnam Bien. An algorithmic approach to fault diagnosis in linear systems. *IEEE trans. Industrial Electronics*, 36(3):313–320, 1989.

[48] K. Kousuke, S. Torsten, S Setsuo, and Y Hisanori. Fault detection of dynamical systems based on model discriminating approach. Technical report, Uppsala University, Department of Technology, P.O. Box 534, Uppsala, Sweden, December 1984.

[49] C. Lassez. Constraint logic programming - a reader. In *Fourth IEEE Symposium on logic programming - San Francisco*, 1987.

[50] C. Lassez, K. McAloon, and R. Yap. Constraint logic programming and option trading. In *Fourth IEEE Symposium on logic programming - San Francisco*, 1987.

[51] J. J. Leary. *Process Monitoring and Fault Diagnosis Methods using Constraint Suspension.* D.Phil. thesis, University of Sussex, 1988.

[52] J. J. Leary and P. J. Gawthrop. Process fault detection using constraint suspension. *Proceedings IEE Pt.D (Special issue on Artificial Intelligence)*, 134(4):264–271, 1987.

[53] Wm Leler. *Constraint Programming Languages.* Addison Wesley, 1988.

[54] W. A. Lodwick. Constraint propagation, relational arithmetic in ai systems and mathematical programs. *Annals of Operations Research*, 21:143–148, 1989.

[55] K. A. Loparo, M. R. Buchner, and K. S. Vasudeva. Leak detection in an experimental heat exchanger process: A multiple model approach. *IEEE Transactions on Automatic Control*, 36:167–177, 1991.

[56] K. F. Martin, P. Thorpe, and S. H. Hoh. Condition monitoring hard faults in a machine tool coolant system. Technical report, University of Wales Institute of Science and Technology, P O Box 23, Cardiff Cf1 3XE, 1988.

[57] K. F. Martin and J. H. Williams. Methodology for condition monitoring of machine tools. In *The First UK Seminar on Condition Monitoring and Diagnostic Engineering Management, Birmingham Polytechnic*, September 1988.

[58] T. Matsuto and S. Shiina. Laboratory test scheduling based on constraint propagation. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 12(3):1334–1335, 1990.

[59] L. A. Mironovski. Functional diagnosis of dynamic systems. a survey. *Automation and Remote Control*, 41:1122–1143, 1980.

[60] P.A.J. Nagy and L.Ljung. System identification using bond graphs. *Proceedings of the First European control Conference*, pages 2564–2569, 1991.

[61] W. R. Nelson. REACTOR: An expert system for diagnosis and treatment of nuclear reactor accidents. In *Proc. of the National Conference on Artificial Intelligence*, page 296, 1982.

[62] H. T. Ng. Model-based, multiple-fault diagnosis of dynamic, continuous physical devices. *IEEE Expert*, December 1991.

[63] R. Patton, P. Frank, and R. Clark. *Fault Diagnosis in Dynamic System.* Prentice Hall International, 1989.

[64] R. J. Patton. Fault detection and diagnosis in aerospace systems using analytical redundancy. *Computing & Control Engineering Journal*, pages 127–136, May 1991.

[65] M. J. Pazzani. Failure-driven learning of fault diagnosis heuristics. *IEEE Transactions and Systems, Man and Cybernetics*, 17:380–394, 1987.

[66] Y. Peng and J. A. Reggia. A probabilistic causal model for diagnostic problem solving part ii:diagnostic strategy. *IEEE Transactions on Systems, Man and Cybernetics*, 17(3):395–406, 1987.

[67] R.B.Martin. To extract the maximum information from imperfect measurements. *British Aerospace technical Magazine*, Sep 89 to Feb 90 (approximately) 1989.

[68] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[69] N. N. Sal'nikov. Problem of the simultaneous estimation of the state vector and parameters of a linear dynamic entity. *Kibernetika i Vychislitel'naya Tekhnika*, 79:5–14, 1988.

[70] E. A. Scarl, J. R. Jamieson, and C. I. Delaune. A fault detection and isolation method applied to liquid oxygen loading for the space shuttle. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 414–416, 1985.

[71] E. A. Scarl, J. R. Jamieson, and C. I. Delaune. Diagnosis and sensor validation through knowledge of structure and function. *IEEE Transactions on Systems, Man and Cybernetics*, 17(3):360–368, 1987.

[72] C. A. Schwartz and H. Ozbay. Recursive identification algorithms as nonlinear systems: Parameter identifiability and controllability. *IFAC Nonlinear Control Systems*, pages 107–112, 1989.

[73] R Sedgewick. *Algorithms.* Addison Wesley, 1988.

[74] Q. Shen and R. Leitch. Diagnosing continuous dynamic systems using qualitative simulation. *Unknown*, pages 1000–1006, 1991.

[75] S. Z. Stefanov. Funcional diagnostics of complex dynamic systems with system interaction. *Cybernetics and Systems: An International Journal*, 20:417–434, 1989.

[76] C. C. Tsui. On the solution to the state failure detection problem. *IEEE Transactions on Automatic Control*, 34(9):1017–1018, 1989.

[77] L. H. Ungar, B. A. Powell, and S. N. Kamens. Adaptive networks for fault diagnosis and process control. *Computers Chem. Engng.*, 14(4-5):561–572, 1990.

[78] A. S. Willsky. A survey of design methods for failure detection systems. *Automatica*, 12:601, 1976.

[79] P. C. Young. Parameter estimation for continuous-time models — a survey. *Automatica*, 17(1):23–39, 1981.

[80] S. K. Yung and D. W. Clarke. Local sensor validation. *Measurement and Control*, 22, June 1989.