DISTRIBUTED DIAGNOSIS FOR STATE-BASED DISCRETE-EVENT SYSTEMS

by

Vengateswaran J. Ravichandran

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science,
Graduate Department of Electrical and Computer Science Engineering,
in the University of Toronto.

To my parents.

Distributed Diagnosis for State-based Discrete-Event Systems

Master of Applied Science Thesis
Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto

by Vengateswaran J. Ravichandran
September 2002

Abstract

This thesis proposes a new method to construct a distributed diagnoser having a coordinated decentralized architecture that consists of local sites communicating with a central unit. This diagnoser is shown to be equivalent, in diagnosability, to the centralized diagnoser of Hashtrudi Zad. The key feature of the proposed distributed diagnoser is that it achieves the same diagnostic performance as the centralized diagnoser, without making any further assumptions. Another important feature of this diagnoser is that it provides the freedom to start the diagnoser at any time during the operation of the system and not necessarily when the system is started. A test case - a practical problem - involving the design of the distributed diagnoser for a nonlinear continuous-time system is also discussed in the thesis.

Contents

List of Tables

List of Figures

List of Symbols

A_1, A_2, A_5	Area of valves open for flow
D	Centralized diagnoser
D_i	i^{th} local observer
F_i	i^{th} failure mode
G	System model, as a finite-state automaton
K_{I_p}, K_{I_w}	Integral constants for pressure and flow, resp.
K_{P_p}, K_{P_w}	Proportional constants for press. & flow, resp.
P_1, P_2, P_{10}	Input pressure variables
$P_{11ref}, P_{21ref}, P_{51ref}$	Pressure references
$P_{11sp}, P_{21sp}, P_{51sp}$	Pressure setpoints
$P_{11s}, P_{21s}, P_{51s}$	Measured pressures
P_{11}, P_{21}, P_{51}	Valve output pressures
P_{12}, P_{22}, P_5	Other pressure variables
$T_1, T_2, T_{12}, T_{22}, T_{51}, T_9$	Temperature variables
W_5	Mass flow into the ECS
$W_{11sp}, W_{21sp}, W_{5sp}$	Flow setpoints
W_{11}, W_{21}, W_{51}	Mass flow through the valves
W_{12s}, W_{22s}, W_{5s}	Measured flows
$W_{12}, W_{13}, W_{22}, W_{24}$	Other mass flow variables
X	State set of G
X_N	Normal state set
X_{F_i}	F_i-mode state set
Y	Output set of G, Event set of D
Y_i	Event set of D_i

Z	State set of D
$\Rightarrow$	Output-adjacency
$\Rightarrow_i$	i-output-adjacency
Σ	Event set of G
$\beta_1, \beta_2, \beta_5$	Angle of opening of valves
δ	Transition function for G
$\hat{x}$	Laplace transform of $x(t); t \geq 0$
κ	Condition map
λ	Output map
λ_i	Output map for D_i
$\mathcal{K}$	Condition set
$\overline{z}$	Global state estimate
$\overline{z}_i$	i^{th} local state estimate
$\psi(\overline{z})$	Reachable output-consistent set for $\overline{z}$
$\psi_i(\overline{z})$	Reachable i-output-consistent set for $\overline{z}$
$\tilde{G}$	Reachability transition system corresp. to G
$\tilde{Z}_i$	State set of D_i
$\widehat{\mathcal{K}}$	Output set of D
ζ	Transition function for D
ζ_i	Transition function for D_i
t_s	Maximum of settling times

Chapter 1

Introduction

Fault detection and isolation are of paramount importance in all systems, especially in aerospace, manufacturing and process industries. As systems become more and more complex, the need for automating the diagnosis has multiplied. Consequently, fault diagnosis has received a lot of attention with schemes based on fault-trees [LGTL85], analytical redundancy [Fra90], expert systems [SW87], models [HCdK92] and on discrete-event systems [SSL$^+$95], [SSL$^+$96], [DLT00], [HZ99], [Su00]. The discussion in this thesis will be limited to fault diagnosis based on discrete-event systems.
In recent years, particular attention has been paid to decentralized architectures ([DLT00], [Su00]) rather than centralized ones; for, as the system becomes increasingly complex and widely distributed (geographically and informationally), it becomes more convenient and effective to utilize several local diagnosers instead of a single centralized diagnoser. Let us first examine a few basic concepts underlying this thesis.

In this thesis, a *failure* (*fault*) refers to a non-permitted deviation in the behaviour of the system or one of its components under supervision for a bounded or unbounded period of time under specified operating conditions. However, in the literature, fault and failure have slightly different meanings [CP99], [Lev95]. A failure is called *permanent* if, after the occurrence of the failure, the system remains in the faulty condition indefinitely. The failure is called *nonpermanent*, otherwise. Fault diagnosis involves *fault detection* followed by *fault isolation*. The former, as the name suggests, is to make a binary decision as to whether or not something has gone wrong. The latter is to locate the source of the failure. Also, we shall limit our discussion to *online passive diagnosis*, that is to say the system under supervision is operational and the fault detection system does not use test inputs for diagnosing failures but relies solely on output signals [HZ99].

In [SSL$^+$95] a method is proposed to construct an event-based model of the system and then construct a diagnoser using state estimations on the plant model via partial observations. [HZ99] proposes a similar method, but based on a state-based model of the system. This method, unlike that of [SSL$^+$95], does not require that the diagnoser and the system be started at the same time. This is a significant advantage when applying the method to practical problems, since it is not easy, in general, to synchronize the plant and the diagnoser at the initial time.

The above two schemes develop a centralized diagnoser. As pointed out earlier, it becomes more convenient to use a distributed diagnoser rather than a centralized diagnoser, as the system becomes more distributed in information. [DLT00] proposes a decentralized architecture with *local observers*, *local diagnosers* and a central *Coordinator* (Figure 1.1). The communication between the local diagnosers and the coordinator is constrained and defined by protocols.

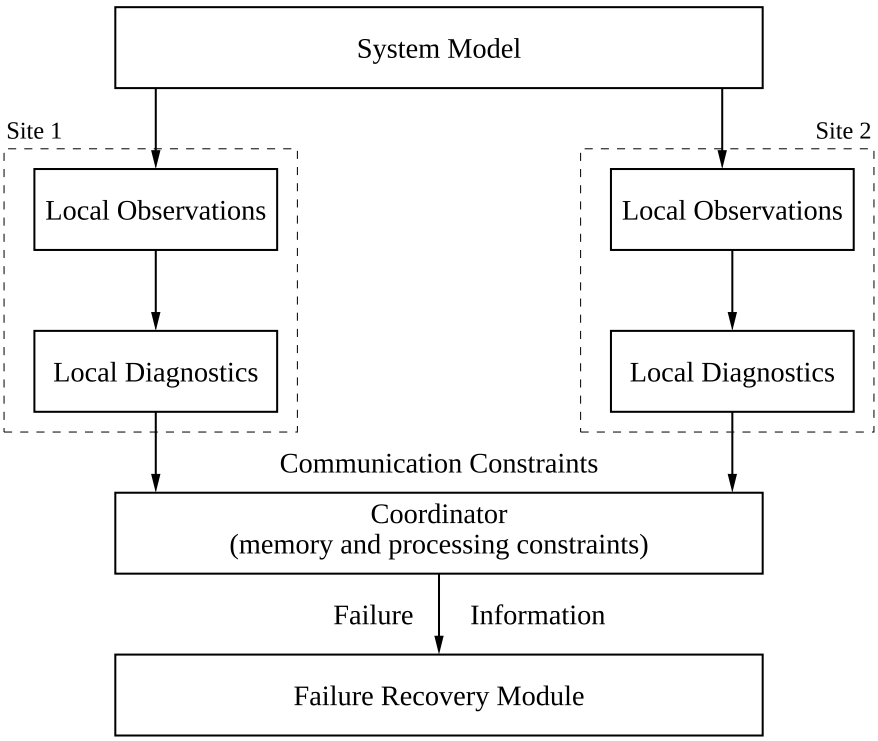

Figure 1.1: Distributed diagnoser of [DLT00]

[Su00] proposes a language-based method to construct a centralized diagnoser, from which the decentralized diagnoser with an appropriate communication mechanism is designed. The central coordinator of [DLT00] is eliminated in favour of communication between the local diagnosers (Figure

1.2). In this design, the communication overhead is reduced by determining what (how much)

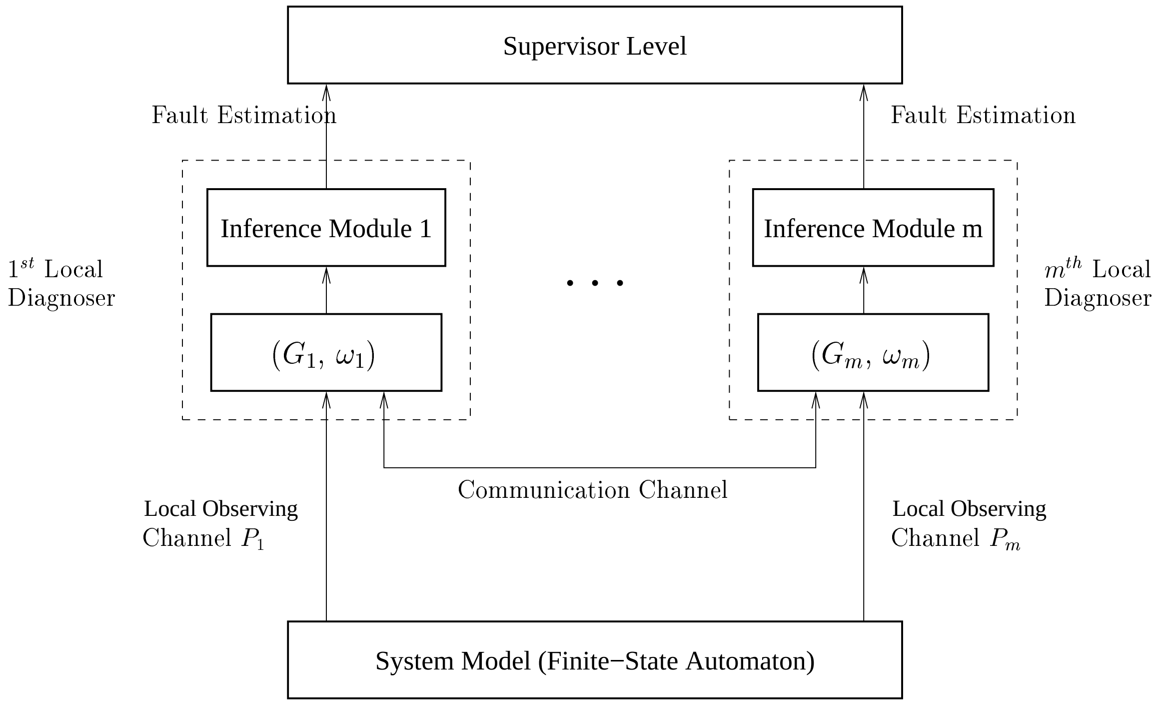

Figure 1.2: Distributed diagnoser of [Su00]

information needs to be communicated and between which local diagnosers.

However, both the above methods require that the system and the diagnoser be initialized at the same instant. In a practical system, it becomes difficult to initialize the diagnoser and the system simultaneously at the same precise instant; this is particularly difficult in systems that are distributed. In either case, it is difficult in the following situation.

Assume that the system fails and the diagnoser detects it. Suppose a fault accommodation procedure initializes a reconfiguration of the system to a new 'mode' i.e., a smooth shift to the new mode while the system still remains online. We then require a corresponding change in the diagnoser in order to keep the system under constant monitoring for faults. It becomes extremely difficult to synchronize the start of the system with that of the diagnoser. It is obvious that it would be much easier to initialize the system in the new mode of operation and then initialize the reconfigured diagnoser.

These conditions warrant that the diagnoser initialization be independent of the initialization of the system. Hence, we propose a new method, based on the model of [HZ99]. In using this model, we overcome the above constraint; i.e., by using the method of this thesis one is able to obtain a distributed diagnoser that need not be initialized simultaneously with the system.

1.1 Thesis Outline

In this thesis, we first describe the framework for diagnosing failures in finite state-automata proposed by [HZ99] in Section 2.1, and in Section 2.2 we describe the centralized diagnoser. In Section 2.4 we propose a new method for obtaining the distributed diagnoser. We then prove that this distributed diagnoser is equivalent to the centralized diagnoser of Section 2.2 under the single-failure scenario, and then extend it to the multiple-failures scenario.

Chapter 3 deals with a practical example, the *Bleed Air System* (BAS). The nonlinear model of the BAS is obtained in Section 3.3. Thereafter, a PI-controller for the system is obtained in Section 3.4 and the results are shown. The faults in the system are described in Section 3.5. In Section 3.6, the DES model of the system is obtained. Section 3.7 describes the centralized diagnoser for the system, while Section 3.8 describes the distributed diagnoser.

Finally, in Chapter 4, a summary of the results and directions for future research are presented.

Chapter 2

Distributed Diagnoser

In this chapter, we first describe the framework for diagnosing failures in finite-state automata proposed in [HZ99]. We also discuss the notions of diagnosability of failures and state the necessary and sufficient conditions for failure diagnosability. Then, the centralized diagnoser explained in [HZ99] is introduced. Expanding on this, we construct a new model for the distributed diagnoser and show that it is equivalent to the centralized diagnoser. We shall use the notation in [HZ99] with slight modifications.

2.1 System model

The system to be diagnosed is modeled as a nondeterministic finite-state Moore automaton

$$G = (X, \Sigma, \delta, x_0, Y, \lambda) \tag{2.1}$$

where X, Σ, and Y are the finite state, event and output sets; x_0 is the initial state; $\delta : X \times \Sigma \to 2^X$ is the transition function, and $\lambda : X \to Y$ is the output map. The above model accounts for both the normal and faulty behaviours of the system. Let $F_1, F_2, \ldots, F_p$ be the p **failure modes** [HZ99]. Each failure mode corresponds to a failure or a set of similar failures. The event set Σ includes the failure events. Under the assumption that only one failure can occur at a time, namely the **single-failure scenario**, the system can only be in one of the following $p + 1$ **conditions**: $N(\text{normal}), F_1, \ldots, F_p$. Let $\mathcal{K} := \{N, F_1, \ldots, F_p\}$, be the condition set of the system. It is assumed that the state set

X can be partitioned according to the condition of the system: $X = X_N \dot{\cup} X_{F_1} \dot{\cup} \ldots \dot{\cup} X_{F_p}$. Define $\kappa : X \mapsto \mathcal{K}$, the **condition map**, such that $\kappa(x) = N$ if $x \in X_N$ and $\kappa(x) = F_i$ if $x \in X_{F_i}$. Extend the definition of κ to any $z \subseteq X$ as follows:

$$\kappa(z) = \cup\{\kappa(x) \mid x \in z\}$$

2.2 Centralized Diagnoser

In the framework of [HZ99], the diagnoser is defined as a finite-state machine that takes the output sequence of the system $y(1)y(2)\ldots y(k)$ as input, and generates as its output an estimate of the condition of the system at the time that $y(k)$ was generated. Specifically, based on the output sequence $y(1)\ldots y(k)$, a set $z(k) \in 2^X - \{\phi\}$ is calculated to which x must belong at the time $y(k)$ was generated. $\kappa(z(k))$ will be the estimate of the system's condition. Upon observing $y(k+1)$, $z(k)$ is updated to $z(k+1)$.

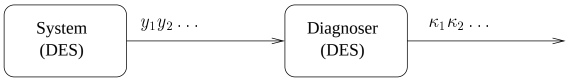

Figure 2.1: System and Diagnoser

Definition 2.1 *For any two states $x, x' \in X$, there is said to be an* **output-consistent path** *from x to x' if $\lambda(x) = \lambda(x')$ and there exist $l \geq 2$, states $x_1, \ldots, x_l$, and events $\sigma_1, \ldots, \sigma_{l-1}$ such that $x_{i+1} \in \delta(x_i, \sigma_i)$ and $\lambda(x_{i+1}) = \lambda(x)$, for all $1 \leq i \leq l-1$, with $x_1 = x$ and $x_l = x'$.* □

Definition 2.2 *(Definition 2.1 in [HZ99]) For any two states $x, x' \in X$, x' is said to be* **output-adjacent** *to x, denoted by $x \Rightarrow x'$, if $\lambda(x) \neq \lambda(x')$ and there exist $l \geq 2$, states $x_1, \ldots, x_{l-1}$, and events $\sigma_1, \ldots, \sigma_{l-1}$ such that $x_{i+1} \in \delta(x_i, \sigma_i)$ and $\lambda(x_i) = \lambda(x)$, for all $1 \leq i \leq l-1$, with $x_1 = x$ and $x_l = x'$.* □

The centralized diagnoser for this system, D, is defined in [HZ99] to be a finite-state Moore machine :

$$D = (Z \cup \{\underline{z}_0\}, Y, \zeta, \underline{z}_0, \widehat{\mathcal{K}}, \kappa)$$

where, $Z \cup \{\underline{z}_0\}, Y, \widehat{\mathcal{K}} \subseteq 2^{\mathcal{K}} - \{\phi\}$ are the state, the event and the output sets; $Z \subseteq 2^X - \{\phi\}$, $\underline{z}_0 := (z_0, 0)$, is the initial state, with $z_0 \in 2^X - \{\phi\}$, and $\zeta : Z \cup \{\underline{z}_0\} \times Y \to Z$ is the transition function (defined below); $\kappa : Z \cup \{\underline{z}_0\} \to \widehat{\mathcal{K}}$ is the output map, with the definition of κ extended according to: $\kappa(\underline{z}_0) := \kappa(z_0)$. The state transition, $z(k+1) := \zeta(z(k), y(k+1))$ is given by:

$$\begin{aligned} z(1) &= z_0 \cap \lambda^{-1}(\{y(1)\}) \\ z(k+1) &= \{x | \lambda(x) = y(k+1) \ \& \ (\exists x' \in z(k) : x' \Rightarrow x)\} \ \ \forall k \geq 1 \end{aligned}$$

Remark 2.1 *$\underline{z}_0$ is considered different from the other states of the diagnoser because the update law ζ is different. Hence, $\underline{z}_0$ is identified with a pair $(z_0, 0)$ rather than a subset of X.*

Remark 2.2 *z_0 contains the estimate of the state of the system at the time the diagnoser is initialized but before readings from sensors are obtained. Since the diagnoser can be initialized at any time while the system is in operation, we do not know the exact state of the system. Hence, usually, $z_0 = X$. However, if the system and the diagnoser are initialized at the same time, then $z_0 = \{x_0\}$. If the system is known to be normal at the time of initialization of the diagnoser, then $z_0 = X_N$.*

For computational efficiency, the information about the output-adjacent states is stored in the **reachability transition system** (**RTS**). The RTS corresponding to G is defined as the transition system $\tilde{G} = (X, R, Y, \lambda)$, which has X, Y, and λ as the state set, output set and the output map. Here, $R \subseteq X \times X$ and $(x_1, x_2) \in R$ if and only if $x_1 \Rightarrow x_2$.

Figure 2.2 shows a system constructed using the above framework and the corresponding diagnoser.

2.3 Diagnosability

In this section, we study the question of whether or not a failure mode F_i can always be detected and isolated. We make the following assumptions:

(A1) Failure modes are permanent

(A2) Single failure scenario

Assumption (A1) implies that once a failure has occurred, the system will stay in that failure mode forever; i.e., there are no temporary failures. Assumption (A2) implies that only one failure mode may occur at a time; that is to say, there are no transitions from the states in X_{F_i} $(i \in \{1, \ldots, p\})$ to any state in $X_N \cup (\cup_{i \neq j} X_{F_j})$ (Figure 2.3).

Definition 2.3 *(Definition 2.2 in [HZ99]) A state z of the diagnoser is called F_i-**certain** if $\kappa(z)$,*

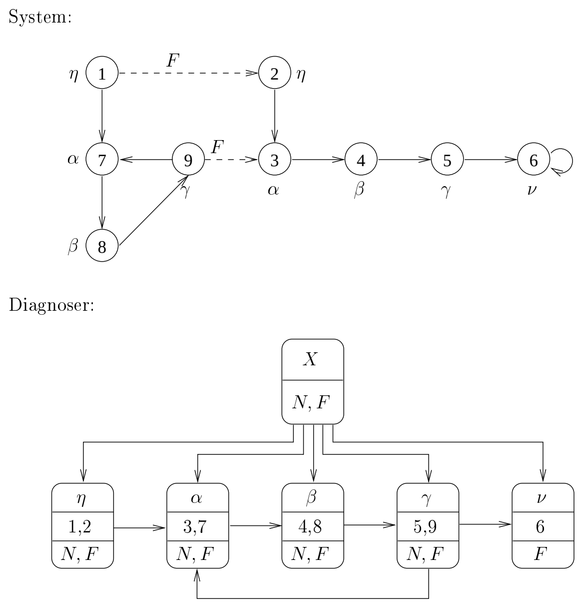

Figure 2.2: A System and the corresponding Diagnoser

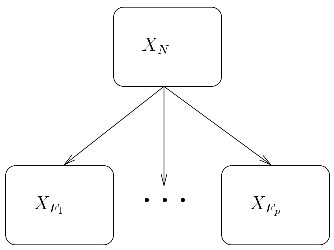

Figure 2.3: Single failure scenario with permanent failure modes

the corresponding estimate of the system's condition, indicates that the failure has occurred.

In the single failure scenario, z is F_i-certain iff $\kappa(z) = \{F_i\}$.

Definition 2.4 *(Definition 2.3 in [HZ99]) A state z of the diagnoser is called F_i-**uncertain** if z is not F_i-certain but $\kappa(z)$ indicates that the failure F_i might have occurred, i.e., occurrence of F_i is consistent with the observed output sequence.* □

In the single failure scenario, z is F_i-uncertain iff $F_i \in \kappa(z)$ but $\kappa(z) \neq \{F_i\}$.
Note that z is not F_i-certain does not mean that it is F_i-uncertain.

Definition 2.5 *(Definition 2.4 in [HZ99]) A permanent failure mode F_i is **diagnosable** if there exists an integer $N_i \geq 0$ such that following both the occurrence of the failure and initialization of* *the diagnoser, F_i can be detected and isolated (i.e., the diagnoser reaches an F_i-certain state) after the occurrence of at most N_i events in the system.* $\square$

Definition 2.6 *(Definition 2.5 in [HZ99]) If the occurrence of a failure mode F_i can be directly concluded from the generation of an output symbol $y \in Y$, then y is called F_i-**indicative**.* $\square$

In the single failure scenario, $y \in Y$ is F_i-indicative iff $\lambda^{-1}(y) \subseteq X_{F_i}$.

Definition 2.7 *(Definition 2.6 in [HZ99]) A cycle [1] of F_i-uncertain states of the diagnoser is called an F_i-**uncertain cycle**.* $\square$

Definition 2.8 *(Definition 2.7 in [HZ99]) Suppose $z^1, \ldots, z^m$ is a cycle of F_i-uncertain states of the diagnoser. The cycle is called F_i-**indeterminate** if there exist $l \geq 1$ and $x_1^j, \ldots, x_l^j \in z^j$, for all $1 \leq j \leq m$ such that $x_k^j \in X_{F_i}$ for all $1 \leq j \leq m$, $1 \leq k \leq l$ and $x_1^1, x_1^2, \ldots, x_1^m, x_2^1, \ldots, x_2^m, \ldots, x_l^1, \ldots, x_l^m$ form a cycle in the RTS. The RTS cycle is called an **underlying fault cycle** of the F_i-indeterminate cycle.* $\square$

Note that not every fault-uncertain cycle is fault-indeterminate. In Figure 2.2, we can see that the diagnoser states $\{3,7\}$, $\{4,8\}$ and $\{5,9\}$ form a cycle of fault-uncertain states; However, this cycle is not fault-indeterminate because the system states 3,4 and 5 do not form a cycle.

Necessary and sufficient conditions for diagnosability of permanent failures in a single-failure scenario, assuming $z_0 = X$, are given by Theorem 2.1.

Theorem 2.1 *(Theorem 2.1 in [HZ99]) Assume the single-failure scenario and $z_0 = X$. A permanent failure F_i is diagnosable if and only if:*

1. *From every $x \in X_{F_i}$, there is at least one transition to another state in X_{F_i} unless $\lambda(x)$ is F_i-indicative,*

2. *there is no cycle in X_{F_i} consisting of states having the same output unless the output symbol is F_i-indicative, and*

3. *there are no F_i indeterminate cycles in the diagnoser.* $\square$

For the reader's convenience, the proof of this theorem (as in [HZ99]) is provided in Appendix A.

[1] A path $x_1 \xrightarrow{\sigma_1} \ldots \xrightarrow{\sigma_{n-1}} x_n$, with $n \geq 2$, is a **cycle** if $x_1 = x_n$.

2.4 Architecture for Distributed Diagnosis

In decentralized systems, the global system information is distributed over several local sites. These local sites may communicate with each other or with a centre. Each distinct information flow leads to a distinct decentralized architecture. Here, we restrict to a coordinated decentralized architecture of the distributed diagnoser shown in Figure 2.4. The distributed diagnoser consists of **local observers**, **local state estimators** and a **global state estimator** and **decision unit**. The communication between the local estimators and the global state estimator is limited to state estimates and an additional bit (as explained later in this section).

We recall that equation 2.1:

$$G = (X, \Sigma, \delta, x_0, Y, \lambda)$$

represents the system under consideration. Furthermore, we recall that Y is the output set of the system. Typically, any $y \in Y$ is the set consisting of the outputs from all the sensors at the instant y is observed.

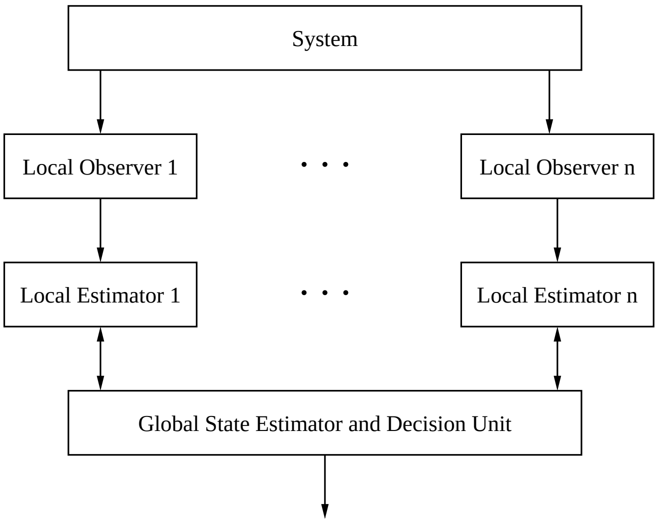

Figure 2.4: Architecture for the distributed diagnoser

The definition of diagnosability under distributed diagnosis remains the same as in the centralized diagnoser (Definition 2.5).

2.4.1 Local Observer

Each site locally observes the output of the system based on the information available locally; typically, the outputs from sensors that are available at that local site. Thus, the output observed by each *local observer* consists of outputs from one or more sensor(s). Let us suppose that there are $n \geq 1$ local observer(s) numbered 1 through n. Then, each local observer i observes the output y_i, the set of outputs from those sensors that are available locally. Y_i, the set of all such y_i, is the output set that is *locally observable*. Using the sequence of outputs it observes, the local observer makes a preliminary estimate of the set of states the system is in.

Remark 2.3 *The output from a sensor can be observable at one or more local observers. In such an instance, the output Y of the system can always be transformed into another equivalent output, $\tilde{Y}$, without affecting the diagnosability and/or any other property of the system G.*
Consider a system $G = (X, \Sigma, \delta, x_0, Y, \lambda)$ with $Y \subseteq S_1 \times \ldots \times S_m$, where S_i is the output set for sensor i. It is assumed here that $S_1, \ldots, S_m$ are the output sets of m distinct sensors. Let n be the number of local observers with locally observed output sets $Y_k \subseteq \prod_{j \in J(k)} S_j$; $J(k) \subseteq \{1, \ldots, m\}$; $k \in \{1, \ldots, n\}$. Define $\tilde{Y} \subseteq \prod_{k=1}^{n} \prod_{j \in J(k)} S_j$ according to:
$\tilde{y} = \big((\prod_{j \in J(1)} s_j), \ldots, (\prod_{j \in J(n)} s_j)\big) \in \tilde{Y}$ if and only if
(a) $(\forall s_j, s_{j'} \in \tilde{y})$ $j = j'$ implies $s_j = s_{j'}$ and (b) $\exists y = (s_1, \ldots, s_m) \in Y$.

Clearly, $\tilde{Y}$ is equivalent to Y in describing the output of the system; any $\tilde{y} \in \tilde{Y}$ only has redundant information that is already contained in an equivalent $y \in Y$. In other words, there exists an injective mapping: $Y \rightarrow \tilde{Y}$ and vice-versa. Hence, from now on it will be assumed that the output Y is the Cartesian product of the outputs observed at the n local sites instead of the m output sensors. In addition, the output map λ is also modified suitably. □

Example 2.1 Consider a system $G = (X, \Sigma, \delta, x_0, Y, \lambda)$ with output $Y \subseteq B \times C \times D$. Let $Y_1 \subseteq B \times C$ and $Y_2 \subseteq C \times D$ be the outputs observed at the local sites 1 and 2 respectively (Figure 2.5). Define $\tilde{Y} \subseteq (B \times C) \times (C \times D)$ according to: $\tilde{y} = (b, c, c', d) \in \tilde{Y}$ if and only if $c = c'$ and $\exists y \in Y : y = (b, c, d)$. Clearly, Y and $\tilde{Y}$ are equivalent. Define $P : Y \rightarrow \tilde{Y}$ such that $\forall y = (b, c, d) \in Y$, $\tilde{y} = Py = (b, c, c, d)$.

Now, define a new system $\tilde{G} = (X, \Sigma, \delta, x_0, \tilde{Y}, \tilde{\lambda})$ with $\tilde{\lambda} : X \mapsto \tilde{Y}$ such that $\tilde{\lambda}(x) = \tilde{y}$ if and only if $\lambda(x) = y$ and $Py = \tilde{y}$. It is clear that G and $\tilde{G}$ are equivalent representations of the system. More importantly, the diagnosability properties of the system G are preserved in $\tilde{G}$. □

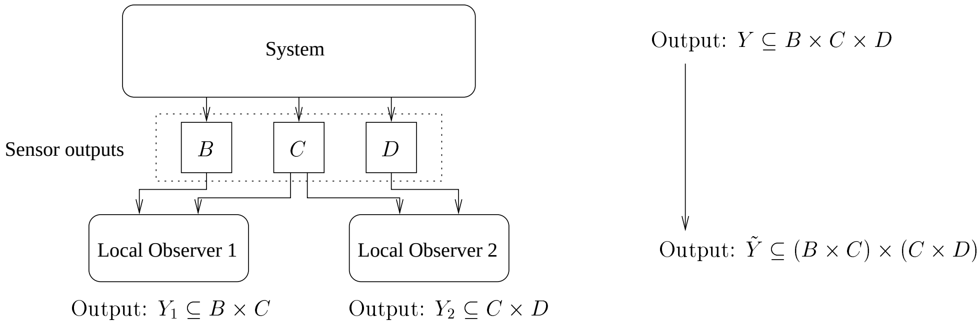

Figure 2.5: Figure for Example 2.1

Therefore, for each local site i, define the natural projection $P_i : Y \to Y_i : (Y_1 \dots Y_n) \to Y_i$.

Definition 2.9 *For any two states $x, x' \in X$, x' is said to be* **i-output-adjacent** *to x, denoted by $x \Rightarrow_i x'$, if $\lambda_i(x) \neq \lambda_i(x')$ and there exist $l \geq 2$, states $x_1, \dots, x_{l-1}$, and events $\sigma_1, \dots, \sigma_{l-1}$ such that $x_{k+1} \in \delta(x_k, \sigma_k)$ and $\lambda_i(x_k) = \lambda_i(x)$, for all $1 \leq k \leq l-1$, with $x_1 = x$ and $x_l = x'$.* $\square$

Definition 2.10 *For any two states $x, x' \in X$, there is said to be an* **i-output-consistent path** *from x to x' if $\lambda_i(x) = \lambda_i(x')$ and there exist $l \geq 2$, states $x_1, \dots, x_l$, and events $\sigma_1, \dots, \sigma_{l-1}$ such that $x_{k+1} \in \delta(x_k, \sigma_k)$ and $\lambda_i(x_{k+1}) = \lambda_i(x)$, for all $1 \leq k \leq l-1$, with $x_1 = x$ and $x_l = x'$.* $\square$

Now, define each local observer as a finite state automaton:

$$D_i = (\tilde{Z}_i \cup \{\tilde{z}_i(0)\}, Y_i, \tilde{z}_i(0), \zeta_i, \lambda_i) \tag{2.2}$$

where,

$$\begin{array}{rcl}
\tilde{Z} & \subseteq & 2^{X \times X} - \{\phi\}, \quad \text{is the state set} \\
\tilde{z}_i(0) & = & \{(x,x) | x \in z_0 \subseteq X\}; \quad \text{is the initial state} \\
\text{with, } z_0 & \in & 2^X - \{\phi\} \\
\zeta_i & : & \tilde{Z}_i \cup \{\tilde{z}_i(0)\} \times Y_i \to \tilde{Z}_i \\
\lambda_i & : & X \mapsto Y_i \quad \text{such that } (\forall x \in X)\ \lambda_i(x) = P_i(\lambda(x)) \\
\tilde{z}_i(k+1) & := & \zeta_i(\tilde{z}_i(k), y_i(k+1)) \\
\tilde{z}_i(1) & = & \tilde{z}_i(0) \cap \{(x,x) | \lambda_i(x) = y_i(1)\}
\end{array}$$

$$\text{for } k \geq 1 \quad \tilde{z}_i(k+1) \quad = \quad \left\{ (x^p, x^c) \;\middle|\; \begin{array}{l} (a)\ \lambda_i(x^c) = y_i(k+1) \\ (b)\ \exists (x'^p, x'^c) \in \tilde{z}_i(k) : x'^c \Rightarrow_i x^c \\ (c)\ (\exists \sigma \in \Sigma)\ \delta(x^p, \sigma) = x^c \\ (d)\ \exists \text{ an } \text{ i} - \text{output} - \text{consistent path from } x'^c \text{ to } x^p \end{array} \right\}$$

In words, $\tilde{z}_i(k+1)$ is the set of states (x^p, x^c) such that $\lambda_i(x^c) = y_i(k+1)$ and x^c is i-output-adjacent to some $x'c$, where $\exists(x'^p, x'^c)$ in $\tilde{z}_i(k)$. x^p is the state that the system reaches immediately before reaching x^c, such that x^p is the last state for which the output is $\lambda(x'^c)$. Note that x^c and x^p are used to denote the current state and the preceding state, respectively.

2.4.2 Local state estimator

Associated with each local observer is a **local state estimator** which estimates the set of all possible states the system could be in using information from the corresponding local observer and the **global state estimator**. The local state estimate for site i after the k^{th} local output is represented by $\overline{z}_i(k)$:

$$\begin{aligned} \overline{z}_i(0) &= \{x \mid (x, x) \in \tilde{z}_i(0)\} \\ \overline{z}_i(1) &= \{x \mid (x, x) \in \tilde{z}_i(1)\} \\ \overline{z}_i(k+1) &= \{x^c \mid \exists x^p (x^p, x^c) \in \tilde{z}_i(k+1) \ \&\ x^p \in \overline{z}(k) \cup \psi(\overline{z}(k))\} \end{aligned}$$

where $\overline{z}(k)$ (note the absence of the suffix i) is the global state estimate (explained in Section 2.4.3) at the instant the k^{th} local observation is made and $\psi : 2^X \to 2^X$ is defined as follows:

$$\psi(\overline{z}) = \{x \in X \mid \exists x' \in \overline{z} \ :\ \exists \text{ an output} - \text{consistent path from } x' \text{ to } x\}$$

Note that the k^{th} local instance of change in output is in general different from the k^{th} global instance of change in output. In Example 2.1, assume that the system goes through a set of states, thus generating the following output sequence: (b, c, d), (b', c, d), (b', c', d). Then, local observer 1 observes the following sequence $(b, c), (b', c), (b', c')$, while local observer 2 observes the sequence $(c, d), (c', d)$. Clearly, the second instance of the local output (at local site 2) corresponds to the third instance, globally. Hence, to avoid this aberration, we define k to be the global instant at which some change in the output occurs and extend the definition of the local state estimate as follows:

$$\begin{aligned}
\overline{z}_i(0) &= \{x \mid (x,x) \in \tilde{z}_i(0)\} \\
\overline{z}_i(1) &= \{x \mid (x,x) \in \tilde{z}_i(1)\} \\
\overline{z}_i(k+1) &= \left\{ \begin{array}{ll} \{x^c \mid \exists x^p (x^p, x^c) \in \tilde{z}_i(k+1) \ \& \ x^p \in \overline{z}(k) \cup \psi(\overline{z}(k))\} & \text{if } \lambda_i(y(k+1)) \neq \lambda_i(y(k)) \\ \psi_i(\overline{z}_i(k)) & \text{otherwise} \end{array} \right\}
\end{aligned}$$

where $\psi_i(\overline{z}) : 2^X \to 2^X$ is defined as follows:

$$\psi_i(\overline{z}) = \left\{ x \in X \;\middle|\; \begin{array}{l} \lambda_i(x) \in \lambda_i(\overline{z}) \ \ \& \\ \exists x' \in \overline{z} \mid \exists \text{ an i} - \text{output} - \text{consistent path from } x \text{ to } x' \end{array} \right\}$$

For computational efficiency, a table consisting of the states x and the corresponding evaluations $\psi_i(x)$ can be constructed for each site i and stored at each local site. For implementation, we use a locally stored binary variable, bs_i. This variable is set to 1 when there is a change in the global output. The local estimate is evaluated and transmitted to the global state estimator once bs_i is set to 1. Upon transmission, the value of bs_i is reset to 0.

2.4.3 Global state estimator and decision unit

The global state estimate, $\overline{z}$ is obtained simply as the intersection of the set of all local state estimates.

$$\overline{z}(k) = \cap_i \overline{z}_i(k)$$

In addition, the global state estimator also maintains a binary variable, bs. This is set to 1 by a local estimator to notify a change in the output it observed. This information is then sent to each local state estimator (for all i, bs_i set to 1), following which the global state estimator receives the local state estimates. Upon receiving the local state estimates, bs is reset to 0.

The estimate of the condition of the system is obtained using the output map on this global state estimate by the decision unit.

2.5 Equivalence of the centralized and the distributed diagnosers

In this section, we prove that the distributed diagnoser above, produces the same diagnostic estimate as the centralized diagnoser of Section 2.2.

Lemma 2.1 *Let $y \in Y$. We claim that*

$$\bigcap_i \lambda_i^{-1}(\{P_i y\}) \subseteq \bigcap_i \lambda^{-1}(\{P_i^{-1}(\{P_i y\}\}))$$

□

Proof

Let $x \in \bigcap_i \lambda_i^{-1}(\{P_i y\})$.

Claim: $x \in \bigcap_i \lambda^{-1}(\{P_i^{-1}(\{P_i y\})\})$

$$\begin{aligned} x &\in \lambda_i^{-1}(\{P_i y\}) \\ \Rightarrow \quad \lambda_i(x) &= P_i y \quad \forall i \\ \Rightarrow \quad (\lambda_1(x), \ldots, \lambda_n(x)) &= (P_1 y, \ldots, P_n y) \\ \Rightarrow \quad \lambda(x) &= y \quad \text{(by defn of } P_i \text{ and } \lambda_i) \\ \Rightarrow \quad x &\in \bigcap_i \lambda^{-1}(\{P_i^{-1}(\{P_i y\})\}) \end{aligned}$$

□

Proposition 2.1 *Let $Z \subseteq 2^X - \{\phi\}$, $z \in Z$ and $y \in Y$. Suppose that*

$$\begin{aligned} z' &= \{x \in z \mid \lambda(x) = y\}, \quad \text{and} \\ \overline{z}' &= \cap_i \{x \in z \mid \lambda_i(x) = P_i y\} \end{aligned}$$

Then $z' = \overline{z}'$. □

Proof

Clearly, $\overline{z}' \supseteq z'$. Therefore it is sufficient to show $\overline{z}' \subseteq z'$.

$$\overline{z}' = \cap_i \{x \in z \mid \lambda_i(x) = P_i y\}$$

$$
\begin{aligned}
&= z \cap \bigcap_i \lambda_i^{-1}(\{P_i y\}) \\
&\subseteq z \cap \bigcap_i \lambda^{-1}(\{P_i^{-1}(\{P_i y\})\}) \quad \text{(by Lemma 2.1)} \\
&= z \cap \lambda^{-1}(\cap_i P_i^{-1}(\{P_i y\})) \\
&= z \cap \lambda^{-1}(\{y\}) \\
&= z'
\end{aligned}
$$

$$\text{Therefore,} \quad \overline{z}' \subseteq z'$$

□

Theorem 2.2 *The state estimate obtained by the centralized diagnoser is the same as the estimate obtained using the decentralized diagnoser.* □

Proof By induction.

$$
\begin{aligned}
\text{By definition,} \quad \overline{z}(0) &= \cap_i \overline{z}_i(0) \\
&= \bigcap_i \{x \mid (x, x) \in \tilde{z}_i(0)\} \\
&= \bigcap_i \{x \mid x \in z_0\} \\
&= z_0
\end{aligned}
$$

$$
\begin{aligned}
\overline{z}(1) &= \cap_i \overline{z}_i(1) \\
&= \bigcap_i \{x \mid (x, x) \in \tilde{z}_i(1)\} \\
&= \bigcap_i \{x \mid (x, x) \in \tilde{z}_i(0)\} \cap \bigcap_i \{x \mid \lambda_i(x) = y_i(1)\} \\
&= z(0) \cap \{x \mid \lambda(x) = y(1)\} \quad \text{(by Proposition 2.1)} \\
&= z(1)
\end{aligned}
$$

Induction step. Assume $\overline{z}(k) = z(k)$. We must show $\overline{z}(k+1) = z(k+1)$.

Case(i): for all i, $y_i(k+1) \neq y_i(k)$

$$
\begin{aligned}
\overline{z}(k+1) &= \cap_i \overline{z}_i(k+1) \\
&= \bigcap_i \{x^c \mid \exists x^p (x^c, x^p) \in \tilde{z}_i(k+1) \ \& \ x^p \in \overline{z}(k) \cup \psi(\overline{z}(k))\}
\end{aligned}
$$

$$
\begin{aligned}
&= \bigcap_i \left\{ x^c \;\middle|\; \begin{array}{l} \exists x^p (x^c, x^p) \in \tilde{z}_i(k+1), \quad \lambda_i(x^c) = y_i(k+1) \\ \exists (x'^p, x'^c) \in \tilde{z}_i(k) \;:\; x'^c \Rightarrow_i x^c \;\&\; \exists \text{ an output} - \text{consistent path from } x'^c \text{ to } x^p \\ x^p \in \overline{z}(k) \cup \psi(\overline{z}(k)) \end{array} \right\} \\
&= \bigcap_i \{x^c \mid \lambda_i(x^c) = y_i(k+1) \;\&\; [\exists x^p \in \overline{z}(k) \;:\; x^p \Rightarrow x^c]\} \\
&= \{x^c \mid \lambda(x^c) = y(k+1) \;\&\; [\exists x^p \in \overline{z}(k) \;:\; x^p \Rightarrow x^c]\} \\
&= z(k+1)
\end{aligned}
$$

Case(ii): Suppose there are n local sites with

$$
\begin{aligned}
y_j(k+1) &= y_j(k) \quad \text{for } j \in J;\ J \subset \{1, \ldots, n\} \\
y_i(k+1) &\neq y_i(k) \quad \text{for } i \in I;\ I = \{1, \ldots, n\} - J \\
\text{then,} \quad \overline{z}(k+1) &= \cap_k \overline{z}_k(k+1) \\
&= \bigcap_{i \in I} \overline{z}_i(k+1) \;\cap\; \bigcap_{j \in J} \psi_j(\overline{z}_j(k+1)) \\
&= \bigcap_{i \in I} \{x^c \mid \exists x^p (x^p, x^c) \in \tilde{z}_i(k+1) \;\&\; [x^p \in \overline{z}(k) \cup \psi(\overline{z}(k))]\} \;\cap\; \bigcap_{j \in J} \{x^c \mid x^c \in \psi_j(\overline{z}_j(k+1))\} \\
&= \bigcap_{i,j} \left\{ x^c \;\middle|\; \begin{array}{l} \lambda_i(x^c) = y_i(k+1) \;\&\; [\exists x^p \in \overline{z}(k) \cup \psi(\overline{z}(k)) \;:\; x^p \Rightarrow x^c], \\ \lambda_j(x^c) = \lambda_j(x^p) = y_j(k) = y_j(k+1) \end{array} \right\} \\
&= \{x^c \mid \lambda(x^c) = y(k+1) \;\&\; [\exists x^p \in \overline{z}(k) \;:\; x^p \Rightarrow x^c]\} \quad (\text{by definition of } \psi) \\
&= z(k+1)
\end{aligned}
$$

Hence, both the central and the distributed diagnosers produce the same diagnosis. □

We have thus extended the theory of diagnosability of systems modeled as state-based DES to the class of decentralized systems. The distributed diagnoser we have obtained is capable of diagnosing all failure types diagnosed by the centralized diagnoser.

2.6 Multiple-failure scenario

In [HZ99] the definition of diagnosability in the case of multiple failures remains the same as that in the single-failure scenario. Hence, the construction of a centralized diagnoser remains the same in both the single-failure case as well as in the case of multiple-failures. As an extension, the distributed diagnoser also remains the same in both the aforesaid cases. However, in the case of multiple failures some modifications to the system (Equation 2.1) are carried out as described below.

For the sake of simplicity, we limit ourselves to only two failure modes - F_1 and F_2. This creates three failure scenarios - F_1, F_2 and F_{12}, with F_{12} denoting the concurrent occurrence of both failure modes F_1 and F_2. Hence the condition set $\mathcal{K} := \{N, F_1, F_2, F_{12}\}$ and the state set is partitioned as $X := X_N \dot{\cup} X_{F_1} \dot{\cup} X_{F_2} \dot{\cup} X_{F_{12}}$. Therefore, the condition map is defined as $\kappa : X \mapsto \mathcal{K}$ according to

$$\kappa(x) = \begin{cases} N & \text{if } x \in X_N \\ F_i & \text{if } x \in X_{F_i};\ (i \in \{1,2\}) \\ F_{12} & \text{if } x \in X_{F_{12}} \end{cases}$$

Also, the condition map is extended to subsets of X as: $\kappa(z) = \{\kappa(x) \mid x \in z\}$ for all $z \subseteq X$.

From Definition 2.3, we obtain that a state $\overline{z}$ of the distributed diagnoser is F_1-certain iff $\kappa(\overline{z}) \subseteq \{F_1, F_{12}\}; \kappa(\overline{z}) \neq \phi$. Similarly, we obtain from Definition 2.4 that a state $\overline{z}$ of the distributed diagnoser is F_1-uncertain iff $\overline{z}$ is not F_1-certain and $\kappa(\overline{z}) \cap \{F_1, F_{12}\} \neq \phi$. And from Definition 2.6, we see that an output y is F_1-indicative iff $\lambda^{-1}(\{y\}) \subseteq X_{F_1} \cup X_{F_{12}}$.

Definition 2.11 *(Definition 2.10 in [HZ99]) Suppose $z^1, \ldots, z^m$ is a cycle of F_i-uncertain states of the diagnoser. The cycle is called F_i-**indeterminate** if there exist $l \geq 1$ and $x_1^j, \ldots, x_l^j \in z^j$, for all $1 \leq j \leq m$ such that $x_k^j \in X_{F_i}$ for all $1 \leq j \leq m$, such that $\{x_k^j | 1 \leq j \leq m, 1 \leq k \leq l\} \subseteq X_{F_1}$ or $\{x_k^j | 1 \leq j \leq m, 1 \leq k \leq l\} \subseteq X_{F_{12}}$, and also $x_1^1, \ldots, x_1^m, x_2^1, \ldots, x_2^m, \ldots, x_l^1, \ldots, x_l^m$ form a cycle in the RTS. The RTS cycle is called an **underlying fault cycle** of the F_1-indeterminate cycle.* □

Now, we have the necessary and sufficient conditions for diagnosability of F_1 .

Theorem 2.3 *(Theorem 2.7 in [HZ99]) Suppose $z_0 = X$. The permanent failure F_1 is diagnosable if and only if:*

1. *From every $x \in X_{F_1} \cup X_{F_{12}}$, there is (at least) one transition to another state in $X_{F_1} \cup X_{F_{12}}$ unless $\lambda(x)$ is F_1-indicative,*
2. *There is no cycle in X_{F_1} or in $X_{F_{12}}$ consisting of states having the same output unless the output symbol is F_1-indicative, and*
3. *There are no F_1-indeterminate cycles in the diagnoser.*

Hence we now have a decentralized diagnoser for the single-failure scenario as well as the multiple-failures scenario, that can diagnose all the faults that can be diagnosed by a centralized diagnoser.

Chapter 3

An application - The Bleed Air System

In this chapter, we design a distributed diagnoser for a practical system using the procedure described in the previous chapter. In order to do this, we first obtain a nonlinear continuous-time model of the system and its controllers. Next, we study the normal and faulty behaviours of the system by running simulations. Using data from the simulations, we construct a discrete-time (DES) model of the system and a centralized diagnoser based on the DES model. We follow it up with a distributed diagnoser. We confirm for selected test cases that the distributed diagnoser diagnoses the failures that are diagnosed by the centralized diagnoser.

Throughout this chapter, we shall use the notation $\hat{x}(s)$ etc., for the Laplace transform of the time function $x(t),\ t \geq 0$.

3.1 The Bleed Air System

In aircraft, the environment inside the compartments where the passengers and the crew are seated is to be maintained at comfortable conditions. For this purpose, in jet aircraft, the compressed air bled out of the compressor stage of the jet engines is conditioned. Conditioning refers to the process by which the pressure and temperature of the exhaust are reduced, and humidity and flow are adjusted, to the appropriate levels. This process of conditioning is taken care of by the *Environment Control System* (ECS). The *Bleed Air System* (BAS) is located between the engine exhaust and the ECS

(Figure 3.1).

The BAS controls the flow rate of the exhaust air bled from the engines into the ECS, according to the demand as well as to prevent excessive bleed of air from the engines, which would in turn affect the amount of air available for the combustion process with obvious repercussions. In addition, the BAS also maintains the pressure of the air going out of it (into the ECS) at standard operating conditions, so that the ECS is not affected by variations in conditions of the exhaust air from the engines. Such variations are common; in particular, during take-off and landing they are rather severe. Thus, the BAS acts as a buffer between the exhaust of the engines and the ECS. The BAS, at the same time, also tries to balance the flow of air from all the engines. This allows for uniform distribution of the load across all engines, thereby preventing overload of any particular engine. This extends the operating life-time of the engines.

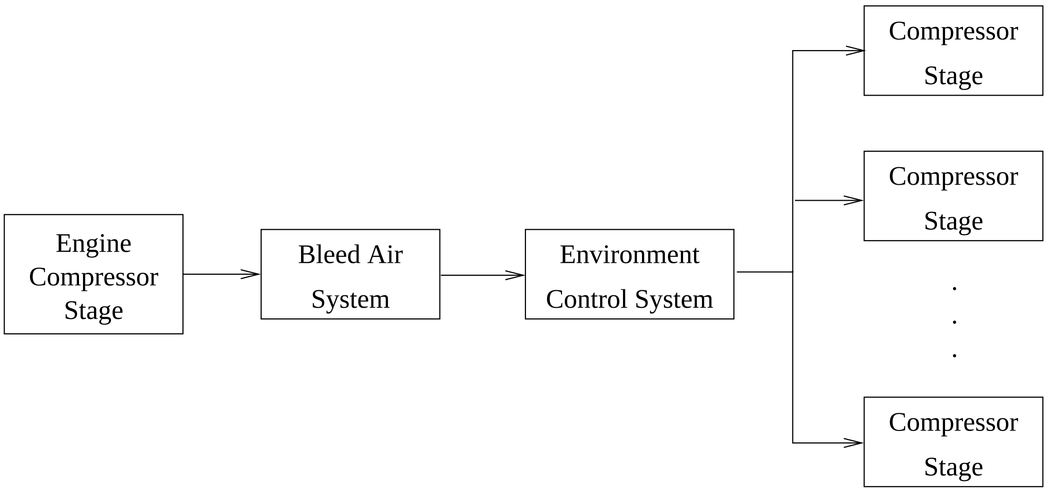

Figure 3.1: Schematic of a Bleed Air System

3.2 Need for diagnosis

As explained before, the BAS acts as a buffer between the exhaust and the ECS, thus requiring its proper functioning for the safety of the passengers as well as the aircraft. An unexpected fault in the BAS on the one hand might propagate down the system and impact the ECS which in turn would affect the passenger and the crew in their compartments; on the other hand, the fault might propagate upstream to affect the combustion process. Obviously, any such fault is undesirable and needs to be detected. Also, since one bleed air system is attached to each engine on both wings of the aircraft, it makes good sense to have a distributed diagnoser rather than a centralized diagnoser.

3.3 Nonlinear Model of the BAS

Figure 3.2 shows a two engine bleed air system. The BAS can essentially be considered to consist of three limbs:

1. Limb 1 - From Engine 1 to point C,
2. Limb 2 - From Engine 2 to point C, and
3. Limb 3 - Point C to the ECS (not including the ECS)

3.3.1 Limb 1

The exhaust air from engine 1 is at temperature T_1 and pressure P_1. A pressure regulator valve, Valve 1, of area A_1 regulates the pressure of the air bled from the engine 1 to a pressure of P_{11} at a flow rate W_{11}. Then, the flow W_{11} is given by [Sha53], [Goo97] as follows :

$$\begin{aligned} W_{11} &= \frac{A_1 P_1}{\sqrt{T_1}} \sqrt{\frac{2\gamma g}{(\gamma-1)R}} \sqrt{\phi_1^{2/\gamma} - \phi_1^{1+\gamma^{-1}}} \qquad (3.1) \\ \text{where} \quad \phi_1 &= P_{11}/P_1 < 0.5283 \qquad \text{(subsonic flow)}, \\ \gamma &= 1.4, \\ R &= 639.6\ in/R^o, \\ g &= 384\ in/s^2, \\ \text{and} \quad A_1 &= \frac{\pi D_1^2}{4}(1-\cos\beta_1) \\ \text{where} \quad D_1 &= \text{the diameter of Valve 1 (in inches), and} \\ \beta_1 &= \text{angle of opening of Valve 1 (in deg.)} \end{aligned}$$

The angle of opening, β_1, for a pressure regulator valve with pressure setpoint, P_{11sp}, and the actual pressure P_{11} is given by [Hon01] :

$$\hat{\beta}_1 = k_1 \frac{\tau_{11}s+1}{s(\tau_{12}s+1)}(\hat{P}_{11sp} - \hat{P}_{11})$$

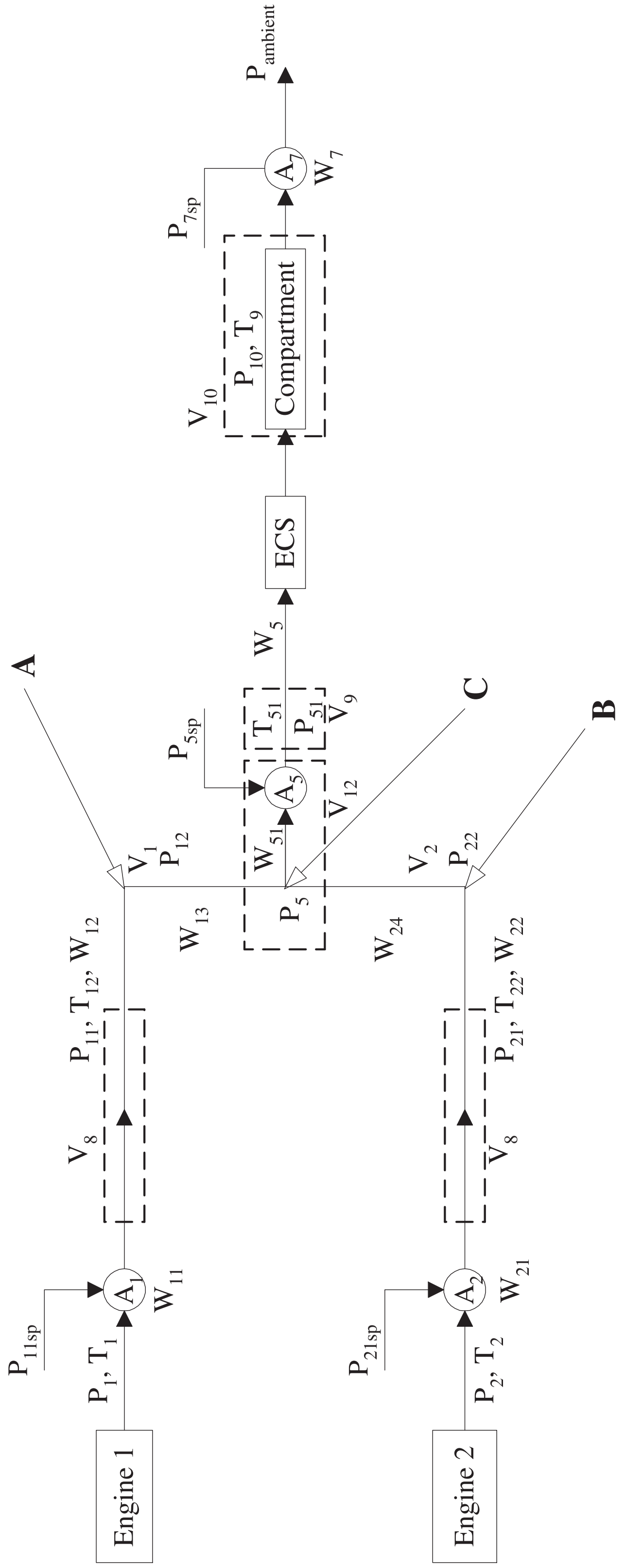

Figure 3.2: A two engine bleed air system (Valve i is represented by 'A_i')

The above equation implies:

$$\begin{aligned}
\tau_{12}\ddot{\beta}_1 + \dot{\beta}_1 &= k_1\tau_{11}(\dot{P}_{11sp} - \dot{P}_{11}) + k_1(P_{11sp} - P_{11}) \ ; \quad \text{initial conditions are } 0 \\
\text{Let} \quad x_1 &:= \tau_{12}\dot{\beta}_1 - k_1\tau_{11}(P_{11sp} - P_{11}) \\
\text{then} \quad \dot{\beta}_1 &= \frac{1}{\tau_{12}}(x_1 + k_1\tau_{11}(P_{11sp} - P_{11})) \qquad (3.2) \\
\text{therefore} \quad \dot{x_1} &= k_1(P_{11sp} - P_{11}) - \dot{\beta}_1 \\
\Rightarrow \quad \dot{x}_1 &= k_1(P_{11sp} - P_{11}) - \frac{1}{\tau_{12}}(x_1 + k_1\tau_{11}(P_{11sp} - P_{11})) \qquad (3.3)
\end{aligned}$$

At point A (Figure 3.2), the pressure is assumed to be P_{12}. The flow into point A, W_{12} is given by [Hon01] :

$$W_{12} = sgn(P_{11} - P_{12})\sqrt{\frac{|P_{11} - P_{12}|}{k_{1112}}} \quad ; \quad k_{1112} \text{ is a constant}$$

However, we replace this sigmoidal function with a linear function based on the range of operation, for the sake of simplicity :

$$W_{12} = \frac{P_{11} - P_{12}}{c_{1112}} \qquad (3.4)$$

where c_{1112} is an appropriate constant for the range of operation.

Similarly, assuming the pressure at point C to be P_5, we obtain W_{13}, the flow into point C from limb 1 as:

$$W_{13} = \frac{P_{12} - P_5}{c_{125}} \qquad (3.5)$$

The volume of the manifold that lies between Valve 1 and the point of measuring P_{11} is V_8. [Buc64] gives the pressure dynamics as follows :

$$\dot{P}_{11} = \frac{\gamma R T_{12}}{V_8}(W_{11} - W_{12}) \qquad (3.6)$$

where T_{12} is the temperature of the above mentioned manifold (Figure 3.2) and is given by

$$T_{12} = T_1 \qquad (3.7)$$

Similar to equation 3.6, we obtain

$$\dot{P}_{12} = \frac{\gamma R T_{12}}{V_1}(W_{12} - W_{13}) \tag{3.8}$$

3.3.2 Limb 2

Limb 2 is similar to Limb 1, *mutatis mutandis.*

The exhaust air from engine 2 is at temperature T_2 and pressure P_2. A pressure regulator valve, Valve 2, of area A_2 regulates the pressure of the air bled from the engine 2 to a pressure of P_{21} at a flow rate W_{21}. Then, the flow W_{21} is given by [Sha53], [Goo97] as follows :

$$\begin{aligned}
W_{21} &= \frac{A_2 P_2}{\sqrt{T_2}}\sqrt{\frac{2\gamma g}{(\gamma-1)R}}\sqrt{\phi_2^{2/\gamma} - \phi_2^{1+\gamma^{-1}}} && (3.9)\\
\text{where} \quad \phi_2 &= P_{21}/P_2 < 0.5283 \qquad \text{(subsonic flow)},\\
\gamma &= 1.4,\\
R &= 639.6\ in/R^o,\\
g &= 384\ in/s^2,\\
\text{and} \quad A_2 &= \frac{\pi D_2^2}{4}(1 - \cos\beta_2)\\
\text{where} \quad D_2 &= \text{ the diameter of Valve 2, and}\\
\beta_2 &= \text{ angle of opening of Valve 2}
\end{aligned}$$

The angle of opening, β_2, for a pressure regulator valve with pressure setpoint, P_{21sp}, and the actual pressure P_{21} is given by [Hon01] :

$$\hat{\beta}_2 = k_2 \frac{\tau_{21}s + 1}{s(\tau_{22}s + 1)}(\hat{P}_{21sp} - \hat{P}_{21})$$

From the above equation, we get (as explained in Section 3.3.1) :

$$\dot{\beta}_2 = \frac{1}{\tau_{22}}(x_2 + k_2\tau_{21}(P_{21sp} - P_{21})) \tag{3.10}$$

$$\dot{x}_2 = k_2(P_{21sp} - P_{21}) - \frac{1}{\tau_{22}}(x_2 + k_2\tau_{21}(P_{21sp} - P_{21})) \tag{3.11}$$

At point B (Figure 3.2), the pressure is assumed to be P_{22}. The the flow into point B, W_{22} is given by [Hon01] :

$$W_{22} = sgn(P_{21} - P_{22})\sqrt{\frac{|P_{21} - P_{22}|}{k_{1112}}} \quad ; \quad k_{1112} \text{ is a constant}$$

As before (Section 3.3.1), we replace this sigmoidal function with a linear function:

$$W_{22} = \frac{P_{21} - P_{22}}{c_{2122}} \tag{3.12}$$

where c_{2122} is an appropriate constant for the range of operation.

Similarly, with the pressure at point C being P_5 (Section 3.3.1), we obtain W_{24}, the flow into point C from limb 2 as:

$$W_{24} = \frac{P_{22} - P_5}{c_{225}} \tag{3.13}$$

The volume of the manifold that lies between Valve 2 and the point of measuring P_{21} is V_8. [Buc64] gives the pressure dynamics as follows:

$$\dot{P}_{21} = \frac{\gamma R T_{22}}{V_8}(W_{21} - W_{22}) \tag{3.14}$$

where T_{22} is the temperature of this manifold (Figure 3.2) and is given by

$$T_{22} = T_2 \tag{3.15}$$

Similar to equation 3.14, we obtain

$$\dot{P}_{22} = \frac{\gamma R T_{22}}{V_2}(W_{22} - W_{24}) \tag{3.16}$$

3.3.3 Limb 3

Limb 3 has similarities with Limb 2 and Limb 1, *mutatis mutandis*.

The flows from Limb 1 and Limb 2 join at point C, leading to a resulting pressure P_5. The control valve, Valve 5, regulates the flow W_{51}, thereby regulating W_5, the flow into the ECS (Figure 3.2). In addition, the valve also controls the pressure P_{51} immediately downstream from it.

The flow W_{51} is given by [Sha53], [Goo97] as follows :

$$\begin{aligned} W_{51} &= \frac{A_5 P_5}{\sqrt{T_5}} \sqrt{\frac{2\gamma g}{(\gamma-1)R}} \sqrt{\phi_5^{2/\gamma} - \phi_5^{1+\gamma^{-1}}} \qquad (3.17) \\ \text{where} \quad \phi_5 &= P_{51}/P_5 < 0.5283 \qquad \text{(subsonic flow)}, \\ \gamma &= 1.4, \\ R &= 639.6 \; in/R^o, \\ g &= 384 \; in/s^2, \\ \text{and} \quad A_5 &= \frac{\pi D_5^2}{4}(1 - \cos\beta_5) \\ \text{where} \quad D_5 &= \text{ the diameter of Valve 5, and} \\ \beta_5 &= \text{ angle of opening of Valve 5} \end{aligned}$$

The angle of opening, β_5, for a pressure regulator valve with pressure setpoint, P_{51sp}, and the actual pressure P_{51} is given by [Hon01] :

$$\hat{\beta}_5 = k_5 \frac{\tau_{51} s + 1}{s(\tau_{52} s + 1)} (\hat{P}_{51sp} - \hat{P}_{51})$$

From the above equation, we get (as explained in Section 3.3.1) :

$$\begin{aligned} \dot{\beta}_5 &= \frac{1}{\tau_{52}}(x_5 + k_5 \tau_{51}(P_{51sp} - P_{51})) \qquad (3.18) \\ \dot{x}_5 &= k_5(P_{51sp} - P_{51}) - \frac{1}{\tau_{52}}(x_5 + k_5 \tau_{51}(P_{51sp} - P_{51})) \qquad (3.19) \end{aligned}$$

P_{10} is the pressure in the passenger compartment that is downstream from the ECS. Then the flow into the ECS, W_5, is given by [Hon01] :

$$W_5 = sgn(P_{51} - P_{10}) \sqrt{\frac{|P_{51} - P_{10}|}{k_{5110}}} \quad ; \quad k_{5110} \text{ is a constant}$$

As before (Section 3.3.1), we replace this sigmoidal function with a linear function:

$$W_5 = \frac{P_{51} - P_{10}}{c_{5110}} \qquad (3.20)$$

where c_{5110} is an appropriate constant for the range of operation.

The temperature at point C (in deg Rankine) is given by :

$$T_5 = \frac{W_{13}(T_{12} - 460) + W_{24}(T_{22} - 460)}{W_{13} + W_{24}} + 460 \tag{3.21}$$

The volume of the manifold that lies between point C and Valve 5 (including the valve) is V_{12}. Also T_5 is the temperature of this manifold (Figure 3.2). Then, [Buc64] gives the pressure dynamics as follows:

$$\dot{P}_5 = \frac{\gamma R T_5}{V_{12}}(W_{13} + W_{24} - W_{51}) \tag{3.22}$$

Similarly, we obtain (Figure 3.2):

$$\dot{P}_{51} = \frac{\gamma R T_{51}}{V_9}(W_{51} - W_5) \tag{3.23}$$

where

$$T_{51} = T_5 \tag{3.24}$$

3.4 Controller

In this section, three controllers are designed to regulate the flow into the ECS, W_5, and to regulate the bleed pressures, P_{11} and P_{21} as well the bleed flows W_{12} and W_{22}.

The flow sensor outputs are given by [Hon01]

$$\hat{W}_{12s} = \frac{1}{.5s+1}\hat{W}_{12} \tag{3.25}$$

$$\hat{W}_{22s} = \frac{1}{.5s+1}\hat{W}_{22} \tag{3.26}$$

$$\hat{W}_{5s} = \frac{1}{.5s+1}\hat{W}_{5} \tag{3.27}$$

The pressure sensor outputs are given by [Hon01]

$$\hat{P}_{11s} = \frac{1}{.1s+1}\hat{P}_{11} \tag{3.28}$$

$$\hat{P}_{21s} = \frac{1}{.1s+1}\hat{P}_{21} \tag{3.29}$$

$$\hat{P}_{51s} = \frac{1}{.1s+1}\hat{P}_{51} \tag{3.30}$$

3.4.1 Operating Point

The operating point values of all the parameters of the system are given in Table 3.1 below [Hon01].

γ	=	1.4	g	=	384 in/s^2
R	=	639.6 in.lbf /^{o}R.lbm			
V_1	=	3600 in^3	V_2	=	10000 in^3
V_8	=	12000 in^3	V_9	=	2000 in^3
V_{12}	=	2000 in^3			
P_{11}	=	60 psi	P_{21}	=	60 psi
k_1	=	0.5 deg.in^2 / lbf.s^2	k_2	=	0.5 deg.in^2 / lbf.s^2
τ_{11}	=	0.02 s	τ_{21}	=	0.02 s
τ_{12}	=	0.01 s	τ_{22}	=	0.01 s
β_1	=	15.7833 deg	β_2	=	15.7833 deg
D_1	=	4 in	D_2	=	4 in
T_1	=	910 ^{o}R	T_2	=	910 ^{o}R
P_1	=	120 psi	P_2	=	120 psi
c_{1112}	=	3 lbf.s / lbm.in^2	c_{2122}	=	3 lbf.s / lbm.in^2
P_{12}	=	57 psi	P_{22}	=	57 psi
c_{125}	=	4 lbf.s / lbm.in^2	c_{225}	=	4 lbf.s / lbm.in^2
P_5	=	53 psi	P_{51}	=	25 psi
k_5	=	0.5 deg.in^2 / lbf.s^2	τ_{51}	=	0.02 s
τ_{52}	=	0.01 s	β_5	=	33.9760 deg
D_5	=	4 in	P_{10}	=	15 psi
c_{5110}	=	5 lbf.s / lbm.in^2			

Table 3.1: Operating Point Values

3.4.2 The Linearized Model

Now, the nonlinear model of Section 3.3 is linearized to obtain the LTI model of the open-loop system about the operating point of Section 3.4.1, on the basis of which the controller for the system will be designed.

Assume the state, input and output vectors of the open loop system are:

$$
\begin{aligned}
x &= [x_1\ \beta_1\ P_{11}\ P_{12}\ x_2\ \beta_2\ P_{21}\ P_{22}\ x_5\ \beta_5\ P_5\ P_{51}]^T \\
u &= [P_1\ T_1\ P_{11sp}\ P_2\ T_2\ P_{21sp}\ P_{51sp}\ P_{10}]^T \\
y &= [\ P_{11}\ P_{12}\ P_{21}\ P_{22}\ P_5\ P_{51}\ W_{11}\ W_{12}\ W_{13}\ W_{21}\ W_{22}\ W_{24}\ W_5\ W_{51}\ \beta_1\ \beta_2\ \beta_5]^T
\end{aligned}
$$

It must be noted here that among the output variables, only $P_{11}, P_{21}, P_{51}, W_{12}, W_{22}$ and W_5 can be measured in the physical system; the other variables are included here solely for the purpose of monitoring by simulation. By linearization of equations 3.1 - 3.24 in Section 3.3 about the operating point of Section 3.4.1, we obtain

$$
\begin{aligned}
\dot{x}(t) &= Ax(t) + Bu(t) \\
y(t) &= Cx(t) + Du(t)
\end{aligned}
$$

with

$$A = \text{columns } 1-6$$

$$\begin{bmatrix}
-100.0000 & 0 & 0.5000 & 0 & 0 & 0 \\
100.0000 & 0 & -1.0000 & 0 & 0 & 0 \\
0 & 8.5496 & -22.6347 & 22.6347 & 0 & 0 \\
0 & 0 & 75.4491 & -132.0359 & 0 & 0 \\
0 & 0 & 0 & 0 & -100.0000 & 0 \\
0 & 0 & 0 & 0 & 0 & 100.0000 \\
0 & 0 & 0 & 0 & 0 & 8.5496 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 101.8563 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$\text{columns } 7-12$$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 56.5868 & 0 \\
0.5000 & 0 & 0 & 0 & 0 & 0 \\
0 & -1.0000 & 0 & 0 & 0 & 0 \\
-22.6347 & 22.6347 & 0 & 0 & 0 & 0 \\
27.1617 & -47.5329 & 0 & 0 & 20.3713 & 0 \\
0 & 0 & -100.0000 & 0 & 0 & 0.5000 \\
0 & 0 & 100.0000 & 0 & 0 & -1.0000 \\
0 & 101.8563 & 0 & -46.5494 & -219.0861 & 0 \\
0 & 0 & 0 & 46.5494 & 15.3735 & -81.4850
\end{bmatrix}$$

$$
B = \begin{bmatrix}
0 & 0 & -0.5000 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 \\
0.5658 & -0.0373 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -0.5000 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 \\
0 & 0 & 0 & 0.5658 & -0.0373 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -0.5000 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 \\
0 & 0.2239 & 0 & 0 & 0.2239 & 0 & 0 & 0 \\
0 & -0.2239 & 0 & 0 & -0.2239 & 0 & 0 & 81.4850
\end{bmatrix}
$$

$$
C = \begin{bmatrix}
0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 \\
0 & 0.1259 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.3333 & -0.3333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.2500 & 0 & 0 & 0 & 0 & 0 & 0 & -0.2500 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.1259 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.3333 & -0.3333 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2500 & 0 & 0 & -0.2500 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2000 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1143 & 0.0377 & 0 \\
0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 & 0 & 0
\end{bmatrix}
$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.0083 & -0.0005 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0083 & -0.0005 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.2000 \\ 0 & -0.0005 & 0 & 0 & -0.0005 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.4.3 Controllers

In this section, three controllers are designed to regulate the flow into the ECS, W_5, and to regulate the bleed pressures, P_{11} and P_{21} as well the bleed flows W_{12} and W_{22} (for flow balancing). For balancing the flows, we adjust the set points for W_{12} and W_{22} - W_{12sp} and W_{22sp} respectively - equal to each other and equal to half of W_{5sp}, the set point for flow W_5. P_{11ref} is the constant *reference* value for P_{11}. It is called *reference* only to differentiate it from P_{11sp}, the set point to Valve 1. P_{11ref} is the constant value we would want P_{11} to reach at steady state; whereas P_{11sp} is the value we set for the valve that controls P_{11} at any given instant. P_{11sp} is different from P_{11ref} because P_{11sp} is also used to control the flow through the valve for flow-balancing (Section 3.1). However, at steady state, $P_{11sp} = P_{11ref}$. Similarly, we define P_{21ref} and P_{51ref} for P_{21} and P_{51} respectively.

For controlling the pressure P_{11} and the flow W_{12} we propose a two input single output cascaded controller

$$\hat{P}_{11sp} = \underbrace{\hat{P}_{11ref} + (K_{P_p} + \frac{K_{I_p}}{s})(\hat{P}_{11ref} - \hat{P}_{11sp})}_{\text{pressure control}} + \underbrace{(K_{P_w} + \frac{K_{I_w}}{s})(\hat{W}_{12sp} - \hat{W}_{12s})}_{\text{flow balance}} \quad (3.31)$$

with $K_{P_w} = 7.0$; $K_{I_w} = 2$; $K_{P_p} = 0.5$; $K_{I_p} = 1$. These values were obtained by tuning using simulations.

Similarly, for controlling the pressure P_{21} and the flow W_{22} the following controller is used

$$\hat{P}_{21sp} = \underbrace{\hat{P}_{21ref} + (K_{P_p} + \frac{K_{I_p}}{s})(\hat{P}_{21ref} - \hat{P}_{21sp})}_{\text{pressure control}} + \underbrace{(K_{P_w} + \frac{K_{I_w}}{s})(\hat{W}_{22sp} - \hat{W}_{22s})}_{\text{flow balance}} \quad (3.32)$$

with $K_{P_w} = 7.0$; $K_{I_w} = 2$; $K_{P_p} = 0.5$; $K_{I_p} = 1$ (obtained by tuning using simulations).

Finally, to control the flow into the ECS, W_5, the following SISO cascaded PI controller is used

$$\hat{P}_{51sp} = \hat{P}_{51ref} + (K_{P_w} + \frac{K_{I_w}}{s})(\hat{W}_{5sp} - \hat{W}_{5s}) \quad (3.33)$$

with $K_{P_w} = 4.0$; $K_{I_w} = 1$ (obtained by tuning using simulations).

3.4.4 Simulations

The nonlinear model of the system (equations 3.1 - 3.24) along with the controller described above (equations 3.31 - 3.33) was implemented as MATLAB code. This program was used to simulate the closed loop response of the system.

First of all the behaviour of the system with nominal values of the input variables was studied. Then, the response of the system for 20% step changes in each individual input, and the combinations thereof, was studied. One such result is shown in Figure 3.3. It shows the output of the system when a 20% step increase in the value of the input P_1 occurs at $t = 45$ s.

3.5 Faults

It is assumed here that:

(A3): No sensor fault occurs

The rationale behind this assumption is that diagnosing sensor faults is independent of the system itself. Furthermore, if this assumption were not made, then it would not be possible to diagnose the

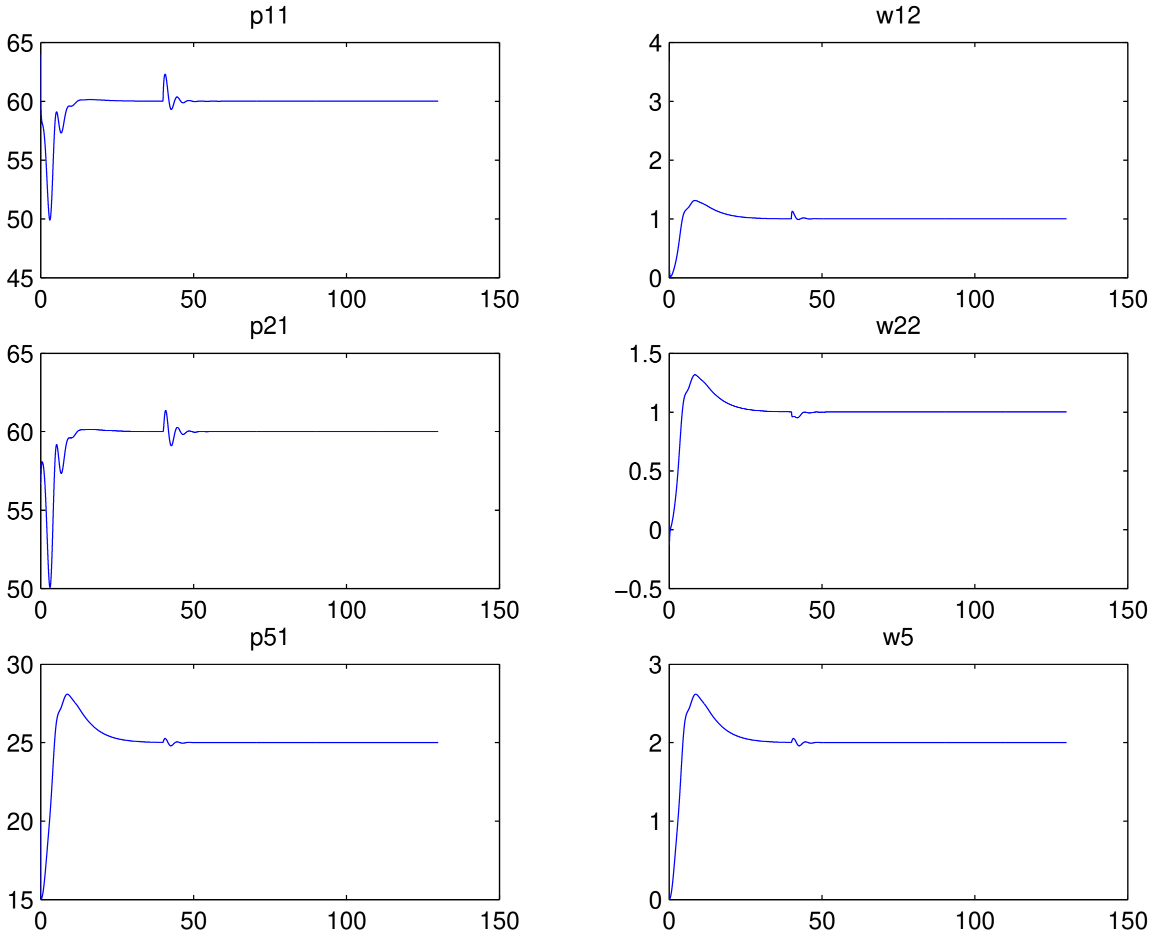

Figure 3.3: Closed loop response for a 20% increase in input P_1
X-axis represents time & Y-axis represents the variable (shown in the title)

faults occurring in the system itself.

Here, the faults considered are the valves–Valve 1, Valve 2 and Valve 5–getting stuck closed or stuck open. They are considered to be abrupt and permanent. Also, only the single failure scenario is considered.

When Valve 1 (or 2 or 5) is stuck closed, then the flow downstream of it, W_{11} (or W_{21} or W_5 respectively), goes to zero and the pressure P_{11} (or P_{21} or P_{51} respectively) drops to equal P_{12} (or P_{22} or P_{10} respectively). When Valve 5 is stuck closed, the pressures P_{12} and P_{22} also start building up until the flows W_{12} and W_{22} drop to zero. As a result, the pressures P_{11} and P_{21} also build up.

When Valve 1 (Valve 2) is stuck open, both the pressure P_{11} (P_{21}) and the pressure downstream, P_{12} (P_{22}), increase. In turn, this results in an increase in the pressure P_{22} (P_{12}). As a result, the flow W_{12} (W_{22}) increases above nominal value; whereas the flow W_{22} (W_{12}) decreases because of the increase in downstream pressure P_{22} (P_{12}).

When Valve 5 gets stuck open, there is an increased flow W_5 and hence a decrease in upstream pressures P_{12} and P_{22}. This causes the flows W_{12} and W_{22} to increase. In turn, the pressure P_{51} increases above nominal value.

Simulations were run using the MATLAB code (Appendix B) to study the faulty behaviour in detail to obtain quantitative information. Figure 3.4 shows the output of the system with Valve 5 stuck closed when $t > 85$ s. Then, simulations were carried out with a white noise of standard deviation 20% of the nominal value of the input added to each input. The resulting disturbance in the system output was studied. Using this information, the tolerance limit for the steady state values was obtained. This tolerance is needed to define *thresholds* when detecting faults.

3.6 DES model of the system

The BAS can be isolated into 3 sites, one local site for each valve, as explained earlier in Section 3.3. Clearly, the set of observable outputs at each site correspond to the pressure and the flow that are immediately related to the local valve; i.e., local site 1 (corresponding to Valve 1) has as locally observable events, the events corresponding to the output from sensors P_{11} and W_{12}. Similarly, events corresponding to output from sensors P_{21} and W_{22} are observable at local site 2 and those from sensors P_{51} and W_5 are observable at the third local site. It was found that the system is in fact perfectly decoupled when the local sites are defined as above. Hence, for experimental purposes, a coupling was induced artificially by defining P_{11} and P_{21} to be observable at local site 1, and W_{12}

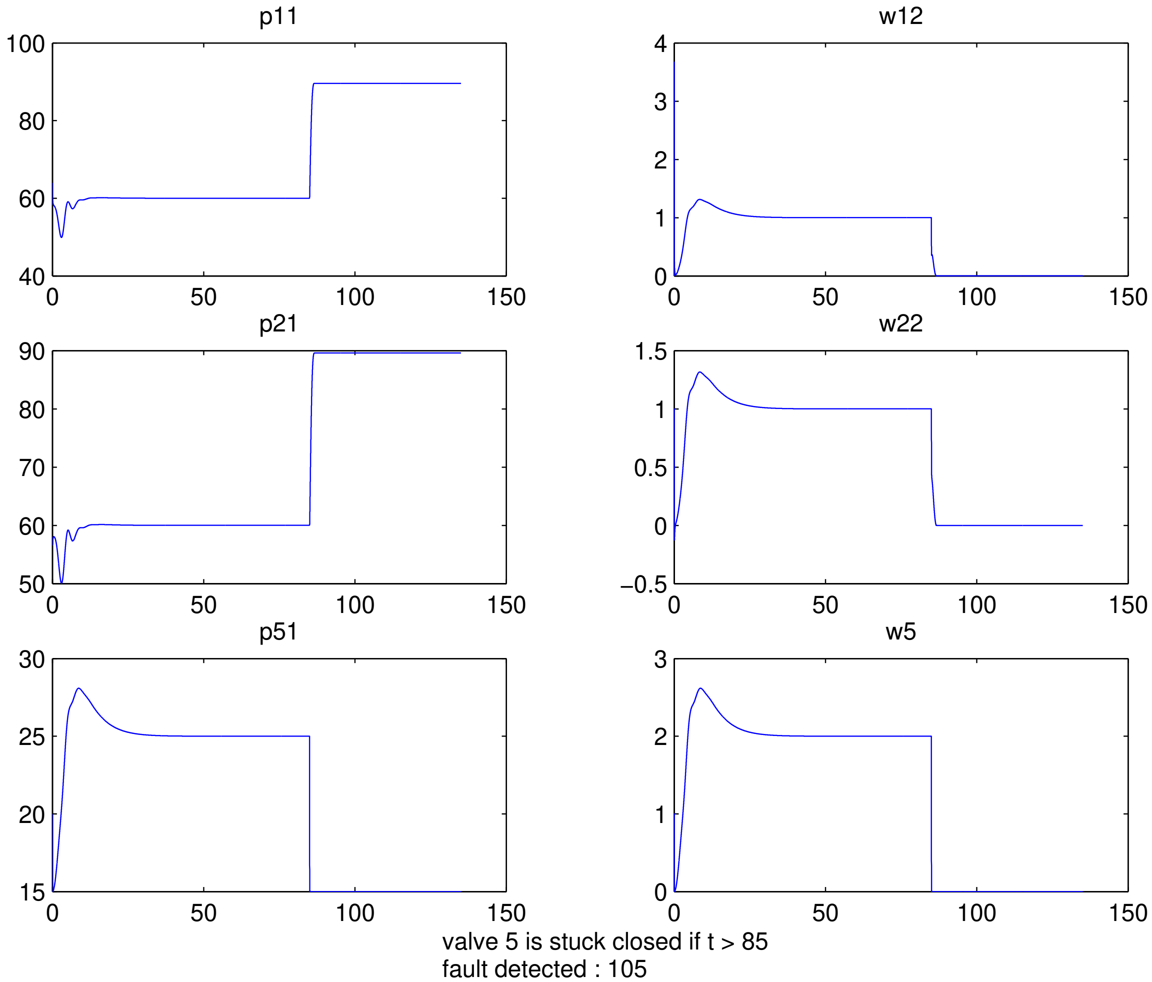

Figure 3.4: Response of the system with Valve 5 stuck closed for $t > 85$
X-axis represents time & Y-axis represents the variable (shown in the title)

and W_{22} to be locally observable at local site 2. Thereby, the failure of Valve 1 or Valve 2 cannot be diagnosed without information from both the local sites 1 and 2. It should be noted here that diagnosis of failure of Valve 5 can still be done independently of the information obtained from local sites 1 and 2; this was left to remain so, intentionally, for the sake of simplicity. Hence from now on, the discussion is limited to local sites 1 and 2, i.e, the system model is limited to the outputs from sensors of P_{11}, P_{21}, W_{12} and W_{22} and the faults corresponding to Valves 1 and 2 only. Henceforth, the symbols P_{11}, P_{21}, W_{12} and W_{22} shall be used to represent interchangeably, the output from the corresponding sensor translated to DES events, the sensors themselves or the appropriate system variables.

Since only the outputs are observable, the system consists of the DES models representing the outputs of the 6 sensors - P_{11}, W_{12}, P_{21}, W_{22}, P_{51}, W_5. In order to translate the output from each sensor to an equivalent DES model, the tolerance limit obtained in Section 3.5 was used to define the following states:

1. state 0: when the output of the sensor is within the tolerance limits

2. state 1: when the output of the sensor is less than the lower threshold

3. state 2: when the output of the sensor is more than the upper threshold

The output of each DES state is thus a discrete measure–below (l), within (e), or above (g) the tolerance limit–of the sensor output the DES represents. Table 3.3 lists the states and their corresponding output. The system components were thus modelled as shown in Figure 3.5.

The synchronous product [Won02] of these 6 components gives the normal behaviour of the system. To include in this the faulty behaviour of the system, the fault DES components were first defined.

If Valve 1 becomes stuck open (event 810 occurs) then P_{11}, P_{21}, W_{12} and W_{22} reach a new equilibrium, with their respective outputs at g, g, g and l. Furthermore, event 810 doesn't occur when their respective outputs are l, l, l and g. Hence, for Valve 1 stuck open, the component DES are shown in Figure 3.6.

If Valve 1 becomes stuck closed (event 812 occurs) then P_{11}, W_{12} and W_{22} reach a new equilibrium, with their respective outputs at l, l and g. Furthermore, event 812 doesn't occur when their respective outputs are g, g and l. Hence, for Valve 1 stuck closed, the component DES are shown in Figure 3.7.

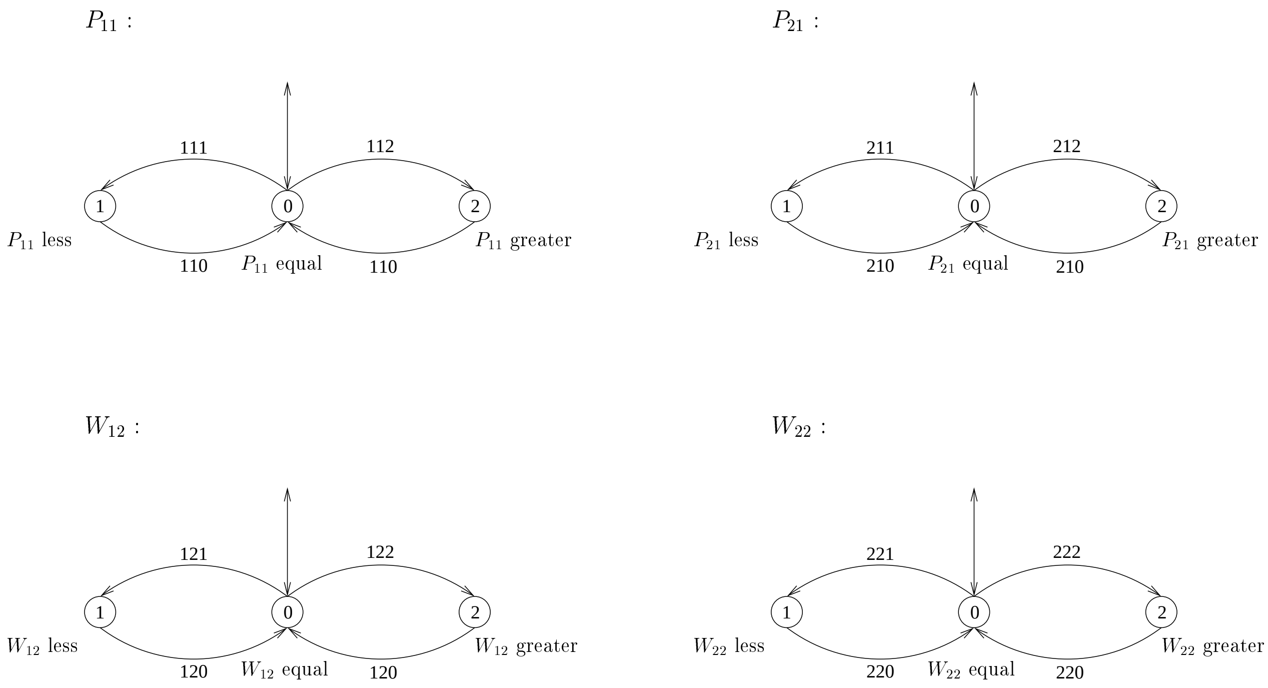

Figure 3.5: DES model of the components of the BAS

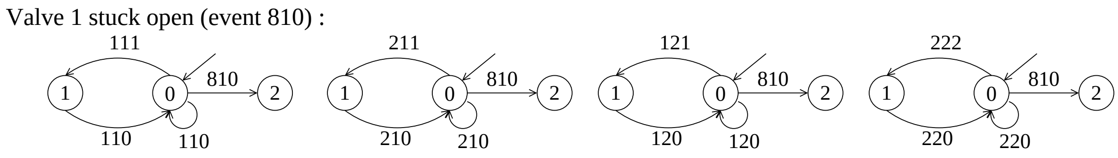

Figure 3.6: Valve 1 stuck open

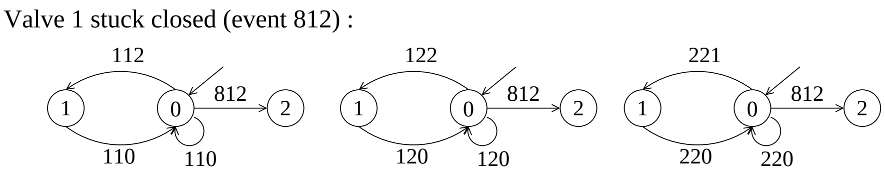

Figure 3.7: Valve 1 stuck closed

Valve 2 stuck open (event 820) :

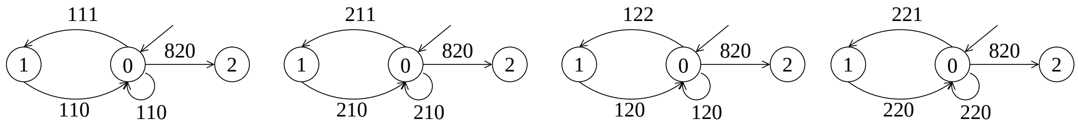

Figure 3.8: Valve 2 stuck open

Valve 2 stuck closed (event 822) :

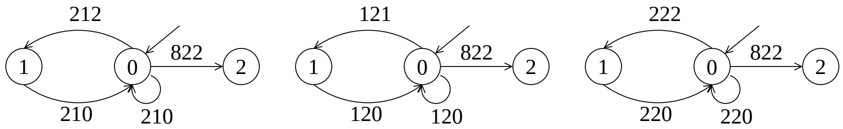

Figure 3.9: Valve 2 stuck closed

If Valve 2 becomes stuck open (event 820 occurs) then P_{11}, P_{21}, W_{12} and W_{22} reach a new equilibrium, with their respective outputs at g, g, l and g. Furthermore, event 820 doesn't occur when their respective outputs are l, l, g and l. Hence, for Valve 2 stuck open, the component DES are shown in Figure 3.8.

If Valve 2 becomes stuck closed (event 822 occurs) then P_{21}, W_{12} and W_{22} reach a new equilibrium, with their respective outputs at l, g and l. Furthermore, event 822 doesn't occur when their respective outputs are g, l and g. Hence, for Valve 2 stuck closed, the component DES are shown in Figure 3.9.

To enforce the single-failure scenario, we have an additional DES component to model the faults, as depicted in Figure 3.10.

Then their (Figures 3.6 - 3.10) synchronous product with the system (Figure 3.5) was obtained. The resulting DES (not shown) of 161 states and 712 events and marker state 0 models the normal

Single failure scenario :

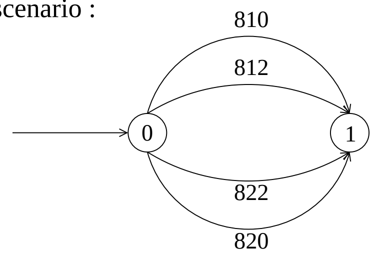

Figure 3.10: Single-failure scenario

as well as the faulty behaviour of the system. The transition table is provided in Table 3.2. The states and their respective outputs are shown in Table 3.3. The condition map of the system is provided in Table 3.4.

Table 3.2: DES model of the system

(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)
(0 , 111 , 1)	(0 , 112 , 2)	(0 , 121 , 3)	(0 , 122 , 4)
(0 , 211 , 5)	(0 , 212 , 6)	(0 , 221 , 7)	(0 , 222 , 8)
(0 , 810 , 9)	(0 , 812 , 10)	(0 , 820 , 11)	(0 , 822 , 12)
(1 , 110 , 0)	(1 , 121 , 13)	(1 , 122 , 14)	(1 , 211 , 15)
(1 , 212 , 16)	(1 , 221 , 17)	(1 , 222 , 18)	(1 , 812 , 19)
(1 , 822 , 20)	(2 , 110 , 0)	(2 , 121 , 21)	(2 , 122 , 22)
(2 , 211 , 23)	(2 , 212 , 24)	(2 , 221 , 25)	(2 , 222 , 26)
(2 , 810 , 27)	(2 , 820 , 28)	(2 , 822 , 29)	(3 , 111 , 13)
(3 , 112 , 21)	(3 , 120 , 0)	(3 , 211 , 30)	(3 , 212 , 31)
(3 , 221 , 32)	(3 , 222 , 33)	(3 , 812 , 34)	(3 , 820 , 35)
(4 , 111 , 14)	(4 , 112 , 22)	(4 , 120 , 0)	(4 , 211 , 36)
(4 , 212 , 37)	(4 , 221 , 38)	(4 , 222 , 39)	(4 , 810 , 40)
(4 , 822 , 41)	(5 , 111 , 15)	(5 , 112 , 23)	(5 , 121 , 30)
(5 , 122 , 36)	(5 , 210 , 0)	(5 , 221 , 42)	(5 , 222 , 43)
(5 , 812 , 44)	(5 , 822 , 45)	(6 , 111 , 16)	(6 , 112 , 24)
(6 , 121 , 31)	(6 , 122 , 37)	(6 , 210 , 0)	(6 , 221 , 46)
(6 , 222 , 47)	(6 , 810 , 48)	(6 , 812 , 49)	(6 , 820 , 50)
(7 , 111 , 17)	(7 , 112 , 25)	(7 , 121 , 32)	(7 , 122 , 38)
(7 , 211 , 42)	(7 , 212 , 46)	(7 , 220 , 0)	(7 , 810 , 51)
(7 , 822 , 52)	(8 , 111 , 18)	(8 , 112 , 26)	(8 , 121 , 33)
(8 , 122 , 39)	(8 , 211 , 43)	(8 , 212 , 47)	(8 , 220 , 0)
(8 , 812 , 53)	(8 , 820 , 54)	(9 , 112 , 27)	(9 , 122 , 40)
(9 , 212 , 48)	(9 , 221 , 51)	(10 , 111 , 19)	(10 , 121 , 34)
(10 , 211 , 44)	(10 , 212 , 49)	(10 , 222 , 53)	(11 , 112 , 28)
(11 , 121 , 35)	(11 , 212 , 50)	(11 , 222 , 54)	(12 , 111 , 20)

continued on next page . . .

. . . continued from previous page

(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)
(12 , 112 , 29)	(12 , 122 , 41)	(12 , 211 , 45)	(12 , 221 , 52)
(13 , 110 , 3)	(13 , 120 , 1)	(13 , 211 , 55)	(13 , 212 , 56)
(13 , 221 , 57)	(13 , 222 , 58)	(13 , 812 , 59)	(14 , 110 , 4)
(14 , 120 , 1)	(14 , 211 , 60)	(14 , 212 , 61)	(14 , 221 , 62)
(14 , 222 , 63)	(14 , 822 , 64)	(15 , 110 , 5)	(15 , 121 , 55)
(15 , 122 , 60)	(15 , 210 , 1)	(15 , 221 , 65)	(15 , 222 , 66)
(15 , 812 , 67)	(15 , 822 , 68)	(16 , 110 , 6)	(16 , 121 , 56)
(16 , 122 , 61)	(16 , 210 , 1)	(16 , 221 , 69)	(16 , 222 , 70)
(16 , 812 , 71)	(17 , 110 , 7)	(17 , 121 , 57)	(17 , 122 , 62)
(17 , 211 , 65)	(17 , 212 , 69)	(17 , 220 , 1)	(17 , 822 , 72)
(18 , 110 , 8)	(18 , 121 , 58)	(18 , 122 , 63)	(18 , 211 , 66)
(18 , 212 , 70)	(18 , 220 , 1)	(18 , 812 , 73)	(19 , 121 , 59)
(19 , 211 , 67)	(19 , 212 , 71)	(19 , 222 , 73)	(20 , 110 , 12)
(20 , 122 , 64)	(20 , 211 , 68)	(20 , 221 , 72)	(21 , 110 , 3)
(21 , 120 , 2)	(21 , 211 , 74)	(21 , 212 , 75)	(21 , 221 , 76)
(21 , 222 , 77)	(21 , 820 , 78)	(22 , 110 , 4)	(22 , 120 , 2)
(22 , 211 , 79)	(22 , 212 , 80)	(22 , 221 , 81)	(22 , 222 , 82)
(22 , 810 , 83)	(22 , 822 , 84)	(23 , 110 , 5)	(23 , 121 , 74)
(23 , 122 , 79)	(23 , 210 , 2)	(23 , 221 , 85)	(23 , 222 , 86)
(23 , 822 , 87)	(24 , 110 , 6)	(24 , 121 , 75)	(24 , 122 , 80)
(24 , 210 , 2)	(24 , 221 , 88)	(24 , 222 , 89)	(24 , 810 , 90)
(24 , 820 , 91)	(25 , 110 , 7)	(25 , 121 , 76)	(25 , 122 , 81)
(25 , 211 , 85)	(25 , 212 , 88)	(25 , 220 , 2)	(25 , 810 , 92)
(25 , 822 , 93)	(26 , 110 , 8)	(26 , 121 , 77)	(26 , 122 , 82)
(26 , 211 , 86)	(26 , 212 , 89)	(26 , 220 , 2)	(26 , 820 , 94)
(27 , 122 , 83)	(27 , 212 , 90)	(27 , 221 , 92)	(28 , 121 , 78)
(28 , 212 , 91)	(28 , 222 , 94)	(29 , 110 , 12)	(29 , 122 , 84)
(29 , 211 , 87)	(29 , 221 , 93)	(30 , 111 , 55)	(30 , 112 , 74)
(30 , 120 , 5)	(30 , 210 , 3)	(30 , 221 , 95)	(30 , 222 , 96)

continued on next page . . .

... *continued from previous page*

(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)
(30 , 812 , 97)	(31 , 111 , 56)	(31 , 112 , 75)	(31 , 120 , 6)
(31 , 210 , 3)	(31 , 221 , 98)	(31 , 222 , 99)	(31 , 812 , 100)
(31 , 820 , 101)	(32 , 111 , 57)	(32 , 112 , 76)	(32 , 120 , 7)
(32 , 211 , 95)	(32 , 212 , 98)	(32 , 220 , 3)	(33 , 111 , 58)
(33 , 112 , 77)	(33 , 120 , 8)	(33 , 211 , 96)	(33 , 212 , 99)
(33 , 220 , 3)	(33 , 812 , 102)	(33 , 820 , 103)	(34 , 111 , 59)
(34 , 211 , 97)	(34 , 212 , 100)	(34 , 222 , 102)	(35 , 112 , 78)
(35 , 212 , 101)	(35 , 222 , 103)	(36 , 111 , 60)	(36 , 112 , 79)
(36 , 120 , 5)	(36 , 210 , 4)	(36 , 221 , 104)	(36 , 222 , 105)
(36 , 822 , 106)	(37 , 111 , 61)	(37 , 112 , 80)	(37 , 120 , 6)
(37 , 210 , 4)	(37 , 221 , 107)	(37 , 222 , 108)	(37 , 810 , 109)
(38 , 111 , 62)	(38 , 112 , 81)	(38 , 120 , 7)	(38 , 211 , 104)
(38 , 212 , 107)	(38 , 220 , 4)	(38 , 810 , 110)	(38 , 822 , 111)
(39 , 111 , 63)	(39 , 112 , 82)	(39 , 120 , 8)	(39 , 211 , 105)
(39 , 212 , 108)	(39 , 220 , 4)	(40 , 112 , 83)	(40 , 212 , 109)
(40 , 221 , 110)	(41 , 111 , 64)	(41 , 112 , 84)	(41 , 211 , 106)
(41 , 221 , 111)	(42 , 111 , 65)	(42 , 112 , 85)	(42 , 121 , 95)
(42 , 122 , 104)	(42 , 210 , 7)	(42 , 220 , 5)	(42 , 822 , 112)
(43 , 111 , 66)	(43 , 112 , 86)	(43 , 121 , 96)	(43 , 122 , 105)
(43 , 210 , 8)	(43 , 220 , 5)	(43 , 812 , 113)	(44 , 111 , 67)
(44 , 121 , 97)	(44 , 210 , 10)	(44 , 222 , 113)	(45 , 111 , 68)
(45 , 112 , 87)	(45 , 122 , 106)	(45 , 221 , 112)	(46 , 111 , 69)
(46 , 112 , 88)	(46 , 121 , 98)	(46 , 122 , 107)	(46 , 210 , 7)
(46 , 220 , 6)	(46 , 810 , 114)	(47 , 111 , 70)	(47 , 112 , 89)
(47 , 121 , 99)	(47 , 122 , 108)	(47 , 210 , 8)	(47 , 220 , 6)
(47 , 812 , 115)	(47 , 820 , 116)	(48 , 112 , 90)	(48 , 122 , 109)
(48 , 221 , 114)	(49 , 111 , 71)	(49 , 121 , 100)	(49 , 210 , 10)
(49 , 222 , 115)	(50 , 112 , 91)	(50 , 121 , 101)	(50 , 222 , 116)
(51 , 112 , 92)	(51 , 122 , 110)	(51 , 212 , 114)	(52 , 111 , 72)

continued on next page ...

... continued from previous page

(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)
(52 , 112 , 93)	(52 , 122 , 111)	(52 , 211 , 112)	(53 , 111 , 73)
(53 , 121 , 102)	(53 , 211 , 113)	(53 , 212 , 115)	(54 , 112 , 94)
(54 , 121 , 103)	(54 , 212 , 116)	(55 , 110 , 30)	(55 , 120 , 15)
(55 , 210 , 13)	(55 , 221 , 117)	(55 , 222 , 118)	(55 , 812 , 119)
(56 , 110 , 31)	(56 , 120 , 16)	(56 , 210 , 13)	(56 , 221 , 120)
(56 , 222 , 121)	(56 , 812 , 122)	(57 , 110 , 32)	(57 , 120 , 17)
(57 , 211 , 117)	(57 , 212 , 120)	(57 , 220 , 13)	(58 , 110 , 33)
(58 , 120 , 18)	(58 , 211 , 118)	(58 , 212 , 121)	(58 , 220 , 13)
(58 , 812 , 123)	(59 , 211 , 119)	(59 , 212 , 122)	(59 , 222 , 123)
(60 , 110 , 36)	(60 , 120 , 15)	(60 , 210 , 14)	(60 , 221 , 124)
(60 , 222 , 125)	(60 , 822 , 126)	(61 , 110 , 37)	(61 , 120 , 16)
(61 , 210 , 14)	(61 , 221 , 127)	(61 , 222 , 128)	(62 , 110 , 38)
(62 , 120 , 17)	(62 , 211 , 124)	(62 , 212 , 127)	(62 , 220 , 14)
(62 , 822 , 129)	(63 , 110 , 39)	(63 , 120 , 18)	(63 , 211 , 125)
(63 , 212 , 128)	(63 , 220 , 14)	(64 , 110 , 41)	(64 , 211 , 126)
(64 , 221 , 129)	(65 , 110 , 42)	(65 , 121 , 117)	(65 , 122 , 124)
(65 , 210 , 17)	(65 , 220 , 15)	(65 , 822 , 130)	(66 , 110 , 43)
(66 , 121 , 118)	(66 , 122 , 125)	(66 , 210 , 18)	(66 , 220 , 15)
(66 , 812 , 131)	(67 , 121 , 119)	(67 , 210 , 19)	(67 , 222 , 131)
(68 , 110 , 45)	(68 , 122 , 126)	(68 , 221 , 130)	(69 , 110 , 46)
(69 , 121 , 120)	(69 , 122 , 127)	(69 , 210 , 17)	(69 , 220 , 16)
(70 , 110 , 47)	(70 , 121 , 121)	(70 , 122 , 128)	(70 , 210 , 18)
(70 , 220 , 16)	(70 , 812 , 132)	(71 , 121 , 122)	(71 , 210 , 19)
(71 , 222 , 132)	(72 , 110 , 52)	(72 , 122 , 129)	(72 , 211 , 130)
(73 , 121 , 123)	(73 , 211 , 131)	(73 , 212 , 132)	(74 , 110 , 30)
(74 , 120 , 23)	(74 , 210 , 21)	(74 , 221 , 133)	(74 , 222 , 134)
(75 , 110 , 31)	(75 , 120 , 24)	(75 , 210 , 21)	(75 , 221 , 135)
(75 , 222 , 136)	(75 , 820 , 137)	(76 , 110 , 32)	(76 , 120 , 25)
(76 , 211 , 133)	(76 , 212 , 135)	(76 , 220 , 21)	(77 , 110 , 33)

continued on next page ...

... continued from previous page

(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)
(77 , 120 , 26)	(77 , 211 , 134)	(77 , 212 , 136)	(77 , 220 , 21)
(77 , 820 , 138)	(78 , 212 , 137)	(78 , 222 , 138)	(79 , 110 , 36)
(79 , 120 , 23)	(79 , 210 , 22)	(79 , 221 , 139)	(79 , 222 , 140)
(79 , 822 , 141)	(80 , 110 , 37)	(80 , 120 , 24)	(80 , 210 , 22)
(80 , 221 , 142)	(80 , 222 , 143)	(80 , 810 , 144)	(81 , 110 , 38)
(81 , 120 , 25)	(81 , 211 , 139)	(81 , 212 , 142)	(81 , 220 , 22)
(81 , 810 , 145)	(81 , 822 , 146)	(82 , 110 , 39)	(82 , 120 , 26)
(82 , 211 , 140)	(82 , 212 , 143)	(82 , 220 , 22)	(83 , 212 , 144)
(83 , 221 , 145)	(84 , 110 , 41)	(84 , 211 , 141)	(84 , 221 , 146)
(85 , 110 , 42)	(85 , 121 , 133)	(85 , 122 , 139)	(85 , 210 , 25)
(85 , 220 , 23)	(85 , 822 , 147)	(86 , 110 , 43)	(86 , 121 , 134)
(86 , 122 , 140)	(86 , 210 , 26)	(86 , 220 , 23)	(87 , 110 , 45)
(87 , 122 , 141)	(87 , 221 , 147)	(88 , 110 , 46)	(88 , 121 , 135)
(88 , 122 , 142)	(88 , 210 , 25)	(88 , 220 , 24)	(88 , 810 , 148)
(89 , 110 , 47)	(89 , 121 , 136)	(89 , 122 , 143)	(89 , 210 , 26)
(89 , 220 , 24)	(89 , 820 , 149)	(90 , 122 , 144)	(90 , 221 , 148)
(91 , 121 , 137)	(91 , 222 , 149)	(92 , 122 , 145)	(92 , 212 , 148)
(93 , 110 , 52)	(93 , 122 , 146)	(93 , 211 , 147)	(94 , 121 , 138)
(94 , 212 , 149)	(95 , 111 , 117)	(95 , 112 , 133)	(95 , 120 , 42)
(95 , 210 , 32)	(95 , 220 , 30)	(96 , 111 , 118)	(96 , 112 , 134)
(96 , 120 , 43)	(96 , 210 , 33)	(96 , 220 , 30)	(96 , 812 , 150)
(97 , 111 , 119)	(97 , 210 , 34)	(97 , 222 , 150)	(98 , 111 , 120)
(98 , 112 , 135)	(98 , 120 , 46)	(98 , 210 , 32)	(98 , 220 , 31)
(99 , 111 , 121)	(99 , 112 , 136)	(99 , 120 , 47)	(99 , 210 , 33)
(99 , 220 , 31)	(99 , 812 , 151)	(99 , 820 , 152)	(100 , 111 , 122)
(100 , 210 , 34)	(100 , 222 , 151)	(101 , 112 , 137)	(101 , 222 , 152)
(102 , 111 , 123)	(102 , 211 , 150)	(102 , 212 , 151)	(103 , 112 , 138)
(103 , 212 , 152)	(104 , 111 , 124)	(104 , 112 , 139)	(104 , 120 , 42)
(104 , 210 , 38)	(104 , 220 , 36)	(104 , 822 , 153)	(105 , 111 , 125)

continued on next page ...

... continued from previous page

(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)
(105 , 112 , 140)	(105 , 120 , 43)	(105 , 210 , 39)	(105 , 220 , 36)
(106 , 111 , 126)	(106 , 112 , 141)	(106 , 221 , 153)	(107 , 111 , 127)
(107 , 112 , 142)	(107 , 120 , 46)	(107 , 210 , 38)	(107 , 220 , 37)
(107 , 810 , 154)	(108 , 111 , 128)	(108 , 112 , 143)	(108 , 120 , 47)
(108 , 210 , 39)	(108 , 220 , 37)	(109 , 112 , 144)	(109 , 221 , 154)
(110 , 112 , 145)	(110 , 212 , 154)	(111 , 111 , 129)	(111 , 112 , 146)
(111 , 211 , 153)	(112 , 111 , 130)	(112 , 112 , 147)	(112 , 122 , 153)
(113 , 111 , 131)	(113 , 121 , 150)	(113 , 210 , 53)	(114 , 112 , 148)
(114 , 122 , 154)	(115 , 111 , 132)	(115 , 121 , 151)	(115 , 210 , 53)
(116 , 112 , 149)	(116 , 121 , 152)	(117 , 110 , 95)	(117 , 120 , 65)
(117 , 210 , 57)	(117 , 220 , 55)	(118 , 110 , 96)	(118 , 120 , 66)
(118 , 210 , 58)	(118 , 220 , 55)	(118 , 812 , 155)	(119 , 210 , 59)
(119 , 222 , 155)	(120 , 110 , 98)	(120 , 120 , 69)	(120 , 210 , 57)
(120 , 220 , 56)	(121 , 110 , 99)	(121 , 120 , 70)	(121 , 210 , 58)
(121 , 220 , 56)	(121 , 812 , 156)	(122 , 210 , 59)	(122 , 222 , 156)
(123 , 211 , 155)	(123 , 212 , 156)	(124 , 110 , 104)	(124 , 120 , 65)
(124 , 210 , 62)	(124 , 220 , 60)	(124 , 822 , 157)	(125 , 110 , 105)
(125 , 120 , 66)	(125 , 210 , 63)	(125 , 220 , 60)	(126 , 110 , 106)
(126 , 221 , 157)	(127 , 110 , 107)	(127 , 120 , 69)	(127 , 210 , 62)
(127 , 220 , 61)	(128 , 110 , 108)	(128 , 120 , 70)	(128 , 210 , 63)
(128 , 220 , 61)	(129 , 110 , 111)	(129 , 211 , 157)	(130 , 110 , 112)
(130 , 122 , 157)	(131 , 121 , 155)	(131 , 210 , 73)	(132 , 121 , 156)
(132 , 210 , 73)	(133 , 110 , 95)	(133 , 120 , 85)	(133 , 210 , 76)
(133 , 220 , 74)	(134 , 110 , 96)	(134 , 120 , 86)	(134 , 210 , 77)
(134 , 220 , 74)	(135 , 110 , 98)	(135 , 120 , 88)	(135 , 210 , 76)
(135 , 220 , 75)	(136 , 110 , 99)	(136 , 120 , 89)	(136 , 210 , 77)
(136 , 220 , 75)	(136 , 820 , 158)	(137 , 222 , 158)	(138 , 212 , 158)
(139 , 110 , 104)	(139 , 120 , 85)	(139 , 210 , 81)	(139 , 220 , 79)
(139 , 822 , 159)	(140 , 110 , 105)	(140 , 120 , 86)	(140 , 210 , 82)

continued on next page ...

... continued from previous page			
(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)	(Exit, Event, In)
(140 , 220 , 79)	(141 , 110 , 106)	(141 , 221 , 159)	(142 , 110 , 107)
(142 , 120 , 88)	(142 , 210 , 81)	(142 , 220 , 80)	(142 , 810 , 160)
(143 , 110 , 108)	(143 , 120 , 89)	(143 , 210 , 82)	(143 , 220 , 80)
(144 , 221 , 160)	(145 , 212 , 160)	(146 , 110 , 111)	(146 , 211 , 159)
(147 , 110 , 112)	(147 , 122 , 159)	(148 , 122 , 160)	(149 , 121 , 158)
(150 , 111 , 155)	(150 , 210 , 102)	(151 , 111 , 156)	(151 , 210 , 102)
(152 , 112 , 158)	(153 , 111 , 157)	(153 , 112 , 159)	(154 , 112 , 160)
(155 , 210 , 123)	(156 , 210 , 123)	(157 , 110 , 153)	(159 , 110 , 153)

Table 3.2: DES model of the system

Table 3.3: Output Map of the system

State - (P_{11}, P_{21}, W_{12}, W_{22})	State - (P_{11}, P_{21}, W_{12}, W_{22})	State - (P_{11}, P_{21}, W_{12}, W_{22})
0 - (e , e , e , e)	1 - (l , e , e , e)	2 - (g , e , e , e)
3 - (e , e , l , e)	4 - (e , e , g , e)	5 - (e , l , e , e)
6 - (e , g , e , e)	7 - (e , e , e , l)	8 - (e , e , e , g)
9 - (e , e , e , e)	10 - (e , e , e , e)	11 - (e , e , e , e)
12 - (e , e , e , e)	13 - (l , e , l , e)	14 - (l , e , g , e)
15 - (l , l , e , e)	16 - (l , g , e , e)	17 - (l , e , e , l)
18 - (l , e , e , g)	19 - (l , e , e , e)	20 - (l , e , e , e)
21 - (g , e , l , e)	22 - (g , e , g , e)	23 - (g , l , e , e)
24 - (g , g , e , e)	25 - (g , e , e , l)	26 - (g , e , e , g)
27 - (g , e , e , e)	28 - (g , e , e , e)	29 - (g , e , e , e)
30 - (e , l , l , e)	31 - (e , g , l , e)	32 - (e , e , l , l)
33 - (e , e , l , g)	34 - (e , e , l , e)	35 - (e , e , l , e)
36 - (e , l , g , e)	37 - (e , g , g , e)	38 - (e , e , g , l)
39 - (e , e , g , g)	40 - (e , e , g , e)	41 - (e , e , g , e)
		continued on next page ...

... continued from previous page

State - (P_{11}, P_{21}, W_{12}, W_{22})	State - (P_{11}, P_{21}, W_{12}, W_{22})	State - (P_{11}, P_{21}, W_{12}, W_{22})
42 - (e , l , e , l)	43 - (e , l , e , g)	44 - (e , l , e , e)
45 - (e , l , e , e)	46 - (e , g , e , l)	47 - (e , g , e , g)
48 - (e , g , e , e)	49 - (e , g , e , e)	50 - (e , g , e , e)
51 - (e , e , e , l)	52 - (e , e , e , l)	53 - (e , e , e , g)
54 - (e , e , e , g)	55 - (l , l , l , e)	56 - (l , g , l , e)
57 - (l , e , l , l)	58 - (l , e , l , g)	59 - (l , e , l , e)
60 - (l , l , g , e)	61 - (l , g , g , e)	62 - (l , e , g , l)
63 - (l , e , g , g)	64 - (l , e , g , e)	65 - (l , l , e , l)
66 - (l , l , e , g)	67 - (l , l , e , e)	68 - (l , l , e , e)
69 - (l , g , e , l)	70 - (l , g , e , g)	71 - (l , g , e , e)
72 - (l , e , e , l)	73 - (l , e , e , g)	74 - (g , l , l , e)
75 - (g , g , l , e)	76 - (g , e , l , l)	77 - (g , e , l , g)
78 - (g , e , l , e)	79 - (g , l , g , e)	80 - (g , g , g , e)
81 - (g , e , g , l)	82 - (g , e , g , g)	83 - (g , e , g , e)
84 - (g , e , g , e)	85 - (g , l , e , l)	86 - (g , l , e , g)
87 - (g , l , e , e)	88 - (g , g , e , l)	89 - (g , g , e , g)
90 - (g , g , e , e)	91 - (g , g , e , e)	92 - (g , e , e , l)
93 - (g , e , e , l)	94 - (g , e , e , g)	95 - (e , l , l , l)
96 - (e , l , l , g)	97 - (e , l , l , e)	98 - (e , g , l , l)
99 - (e , g , l , g)	100 - (e , g , l , e)	101 - (e , g , l , e)
102 - (e , e , l , g)	103 - (e , e , l , g)	104 - (e , l , g , l)
105 - (e , l , g , g)	106 - (e , l , g , e)	107 - (e , g , g , l)
108 - (e , g , g , g)	109 - (e , g , g , e)	110 - (e , e , g , l)
111 - (e , e , g , l)	112 - (e , l , e , l)	113 - (e , l , e , g)
114 - (e , g , e , l)	115 - (e , g , e , g)	116 - (e , g , e , g)
117 - (l , l , l , l)	118 - (l , l , l , g)	119 - (l , l , l , e)
120 - (l , g , l , l)	121 - (l , g , l , g)	122 - (l , g , l , e)
123 - (l , e , l , g)	124 - (l , l , g , l)	125 - (l , l , g , g)
126 - (l , l , g , e)	127 - (l , g , g , l)	128 - (l , g , g , g)

continued on next page ...

... continued from previous page

State - (P_{11}, P_{21}, W_{12}, W_{22})	State - (P_{11}, P_{21}, W_{12}, W_{22})	State - (P_{11}, P_{21}, W_{12}, W_{22})
129 - (l , e , g , l)	130 - (l , l , e , l)	131 - (l , l , e , g)
132 - (l , g , e , g)	133 - (g , l , l , l)	134 - (g , l , l , g)
135 - (g , g , l , l)	136 - (g , g , l , g)	137 - (g , g , l , e)
138 - (g , e , l , g)	139 - (g , l , g , l)	140 - (g , l , g , g)
141 - (g , l , g , e)	142 - (g , g , g , l)	143 - (g , g , g , g)
144 - (g , g , g , e)	145 - (g , e , g , l)	146 - (g , e , g , l)
147 - (g , l , e , l)	148 - (g , g , e , l)	149 - (g , g , e , g)
150 - (e , l , l , g)	151 - (e , g , l , g)	152 - (e , g , l , g)
153 - (e , l , g , l)	154 - (e , g , g , l)	155 - (l , l , l , g)
156 - (l , g , l , g)	157 - (l , l , g , l)	158 - (g , g , l , g)
159 - (g , l , g , l)	160 - (g , g , g , l)	

Table 3.3: Output Map of the system

Table 3.4: Condition map of the system

State - Condition	State - Condition	State - Condition	State - Condition	State - Condition
0 - Normal	1 - Normal	2 - Normal	3 - Normal	4 - Normal
5 - Normal	6 - Normal	7 - Normal	8 - Normal	9 - V1so
10 - V1sc	11 - V2so	12 - V2sc	13 - Normal	14 - Normal
15 - Normal	16 - Normal	17 - Normal	18 - Normal	19 - V1sc
20 - V2sc	21 - Normal	22 - Normal	23 - Normal	24 - Normal
25 - Normal	26 - Normal	27 - V1so	28 - V2so	29 - V2sc
30 - Normal	31 - Normal	32 - Normal	33 - Normal	34 - V1sc
35 - V2so	36 - Normal	37 - Normal	38 - Normal	39 - Normal
40 - V1so	41 - V2sc	42 - Normal	43 - Normal	44 - V1sc
45 - V2sc	46 - Normal	47 - Normal	48 - V1so	49 - V1sc
50 - V2so	51 - V1so	52 - V2sc	53 - V1sc	54 - V2so

continued on next page ...

... continued from previous page				
State - Condition	State - Condition	State - Condition	State - Condition	State - Condition
55 - Normal	56 - Normal	57 - Normal	58 - Normal	59 - V1sc
60 - Normal	61 - Normal	62 - Normal	63 - Normal	64 - V2sc
65 - Normal	66 - Normal	67 - V1sc	68 - V2sc	69 - Normal
70 - Normal	71 - V1sc	72 - V2sc	73 - V1sc	74 - Normal
75 - Normal	76 - Normal	77 - Normal	78 - V2so	79 - Normal
80 - Normal	81 - Normal	82 - Normal	83 - V1so	84 - V2sc
85 - Normal	86 - Normal	87 - V2sc	88 - Normal	89 - Normal
90 - V1so	91 - V2so	92 - V1so	93 - V2sc	94 - V2so
95 - Normal	96 - Normal	97 - V1sc	98 - Normal	99 - Normal
100 - V1sc	101 - V2so	102 - V1sc	103 - V2so	104 - Normal
105 - Normal	106 - V2sc	107 - Normal	108 - Normal	109 - V1so
110 - V1so	111 - V2sc	112 - V2sc	113 - V1sc	114 - V1so
115 - V1sc	116 - V2so	117 - Normal	118 - Normal	119 - V1sc
120 - Normal	121 - Normal	122 - V1sc	123 - V1sc	124 - Normal
125 - Normal	126 - V2sc	127 - Normal	128 - Normal	129 - V2sc
130 - V2sc	131 - V1sc	132 - V1sc	133 - Normal	134 - Normal
135 - Normal	136 - Normal	137 - V2so	138 - V2so	139 - Normal
140 - Normal	141 - V2sc	142 - Normal	143 - Normal	144 - V1so
145 - V1so	146 - V2sc	147 - V2sc	148 - V1so	149 - V2so
150 - V1sc	151 - V1sc	152 - V2so	153 - V2sc	154 - V1so
155 - V1sc	156 - V1sc	157 - V2sc	158 - V2so	159 - V2sc
160 - V1so				

Table 3.4: Condition map of the system

3.7 Centralized Diagnoser

The system as such is not diagnosable. This is because the system reaches states that have the same output as that of a faulty state even under normal conditions. The only difference is that after a failure occurs, the system is restricted to a particular set of states, whereas under normal conditions, the system returns back to its steady state value. For instance, when Valve 1 gets stuck closed, P_{11}, W_{12} and W_{22} reach a new equilibrium point, corresponding to the fault - at outputs l, l and g respectively. The system could reach the very same state even under normal conditions, say due to a disturbance in one of the inputs (P_1, in the present case). However, under normal conditions, the system returns to its nominal state. This means that the diagnoser can detect a failure only by looking at how long the system remains in a state. Thus, by including such timing information, the system can be made diagnosable.
Hence, it is assumed that
(A4) The system has sufficient time to return to steady state before the occurrence of another disturbance.

From the simulations, the settling time, i.e., minimum time taken by the system to reach and remain within 10% of its equilibrium value, is obtained. The maximum of all such settling times was found to be $t_s = 15$s. Hence the diagnoser essentially checks to see if the system stays in the same faulty state for time t_s. If it does, then under the assumption (A4), it is presumed that a failure has occurred. In addition, assumption (A4) makes it sufficient to observe the output of the system only twice every t_s seconds (by Nyquist's criterion). However, the frequency of observation could be increased and is limited only by the processing power available.

The diagnoser was implemented as MATLAB code (Appendix B) and simulations were run to check its validity. Figure 3.4 shows the system output with Valve 5 stuck closed if $t > 85$ s. The diagnoser detects the fault at $t = 105$ s.

3.8 Distributed Diagnoser

In this section, a distributed diagnoser for the system is constructed using the procedure described in Chapter 2. The distributed diagnoser gives the same estimate as the central diagnoser above.

3.8.1 Local Observer

As explained in Section 3.6, local site 1 consists of events corresponding to outputs of sensors P_{11} and P_{21}, while local site 2 consists of events corresponding to outputs of sensors W_{12} and W_{22}; i.e., $Y_1 = P_{11} \times P_{21}$ and $Y_2 = W_{12} \times W_{22}$.

Hence, the models of the local observers 1 and 2 are obtained from the system model (Section 3.6) using the procedure described in Section 2.4.1.

3.8.2 Local Estimators

The local state estimator, as explained in Section 2.4.2, simply computes the local estimate based on the observation. As had been pointed out earlier, for the sake of computational efficiency, a table consisting of $\psi_i(x)$ (Section 2.4.2) for each state x is calculated apriori and stored. Tables 3.5 and 3.6 represent the above. Note that in each table only states for which $\psi_i(x)$ are nonempty are listed.

Table 3.5: Table of $\psi_1(x)$

State - $\psi_1(x)$	State - $\psi_1(x)$	State - $\psi_1(x)$
0 - 3,4,7,8,9,10,11,12	1 - 13,14,17,18,19,20	2 - 21,22,25,26,27,28,29
3 - 0,32,33,34,35	4 - 0,38,39,40,41	5 - 30,36,42,43,44,45
6 - 31,37,46,47,48,49,50	7 - 0,32,38,51,52	8 - 0,33,39,53,54
9 - 40,51	10 - 34,53	11 - 35,54
12 - 41,52	13 - 1,57,53,59	14 - 1,62,63,64
15 - 55,60,65,66,67,68	16 - 56,61,69,70,71	17 - 1,57,62,72
18 - 1,58,63,73	19 - 59,73	20 - 64,72
21 - 2,76,77,78	22 - 2,81,82,83,84	23 - 74,79,85,86,87
24 - 75,80,88,89,90,91	25 - 2,76,81,92,93	26 - 2,77,82,94
27 - 83,92	28 - 78,94	29 - 84,93
30 - 5,95,96,97	31 - 6,98,99,100,101	32 - 3,7
33 - 3,8,102,103	34 - 102	35 - 103
36 - 5,104,105,106	37 - 6,107,108,109	38 - 4,7,110,111
39 - 4,8	40 - 110	41 - 111
42 - 5,95,104,112	43 - 5,96,105,113	44 - 97

continued on next page ...

... continued from previous page

State - $\psi_1(x)$	State - $\psi_1(x)$	State - $\psi_1(x)$
45 - 106,112	46 - 6,98,107,114	47 - 6,99,108,115,116
48 - 109,114	49 - 100,115	50 - 101,116
51 - 110	52 - 111	53 - 102
54 - 103	55 - 15,117,118,119	56 - 16,120,121,122
57 - 13,17	58 - 13,18,123	59 - 123
60 - 15,124,125,126	61 - 16,124,128	62 - 14,17,129
63 - 14,18	64 - 129	65 - 15,117,124,130
66 - 15,118,125,131	67 - 119,131	68 - 126,130
69 - 16,120,127	70 - 16,121,128,132	71 - 122,132
72 - 129	73 - 123	74 - 23,133,134
75 - 24,135,136,137	76 - 21,25	77 - 21,26,138
78 - 138	79 - 23,139,140,141	80 - 24,142,143,144
81 - 22,25,145,146	82 - 22,26	83 - 145
84 - 146	85 - 12,133,139,147	86 - 23,134,140
87 - 141,147	88 - 24,135,142,148	89 - 24,136,143,149
90 - 144,148	91 - 137,149	92 - 145
93 - 146	94 - 138	95 - 30,42
96 - 30,43,150	97 - 150	98 - 31,46
99 - 31,47,151,152	100 - 151	101 - 152
104 - 36,42,153	105 - 36,43	106 - 153
107 - 37,46,154	108 - 37,47	109 - 154
112 - 153	113 - 150	114 - 151
115 - 152	117 - 55,65	118 - 55,66
119 - 155	120 - 56,69	121 - 56,70,156
122 - 156	124 - 60,65,157	125 - 60,66
126 - 157	127 - 61,69	128 - 61,70
130 - 157	131 - 155	132 - 156
133 - 74,85	134 - 74,86	135 - 75,88
136 - 75,89,158	137 - 158	139 - 79,85,159

continued on next page ...

... continued from previous page

State - $\psi_1(x)$	State - $\psi_1(x)$	State - $\psi_1(x)$
140 - 79,86	141 - 159	142 - 88,160
143 - 80,89	144 - 160	147 - 159
148 - 160	149 - 158	

Table 3.5: Table of $\psi_1(x)$

Table 3.6: Table of $\psi_2(x)$

State - $\psi_2(x)$	State - $\psi_2(x)$	State - $\psi_2(x)$
0 - 1,2,5,6,9,10,11,12	1 - 0,15,16,19,20	2 - 0,23,24,27,28,29
3 - 13,21,30,31,34,35	4 - 14,22,36,37,40,41	5 - 0,15,23,44,45
6 - 0,16,24,48,49,50	7 - 17,25,42,46,51,52	8 - 18,26,43,47,53,54
9 - 27,48	10 - 19,44,49	11 - 28,50
12 - 20,29,45	13 - 3,55,56,59	14 - 4,60,61,64
15 - 1,5,67,68	16 - 1,6,71	17 - 7,65,69,72
18 - 8,66,70,73	19 - 67,71	20 - 12,68
21 - 3,74,75,78	22 - 4,79,80,83,84	23 - 2,5,87
24 - 2,6,90,91	25 - 7,85,88,92,93	26 - 8,86,89,94
27 - 90	28 - 91	29 - 12,87
30 - 3,55,74,97	31 - 3,56,75,100,101	32 - 57,76,95,98
33 - 58,77,96,99,102,103	34 - 59,97,100	35 - 78,101
36 - 4,60,79,106	37 - 4,61,80,109	38 - 62,81,104,107,110,111
39 - 63,82,105,108	40 - 83,109	41 - 64,84,106
42 - 7,65,85,112	43 - 8,66,86,113	44 - 10,67
45 - 68,87	46 - 7,69,88,114	47 - 8,70,89,115,116
48 - 90	49 - 10,71	50 - 91
51 - 92,114	52 - 72,93,112	53 - 73,113,115
54 - 94,116	55 - 13,30,119	56 - 13,31,122

continued on next page ...

... continued from previous page

State - $\psi_2(x)$	State - $\psi_2(x)$	State - $\psi_2(x)$
57 - 32,117,120	58 - 33,118,121,123	59 - 119,122
60 - 14,36,126	61 - 14,37	62 - 38,124,127,129
63 - 39,125,128	64 - 41,126	65 - 17,42,130
66 - 18,43,131	67 - 19	68 - 45
69 - 17,46	70 - 18,47,132	71 - 19
72 - 52,130	73 - 131,132	74 - 21,30
75 - 21,31,137	76 - 32,133,135	77 - 33,134,136,138
78 - 137	79 - 22,36,141	80 - 22,37,144
81 - 38,139,142,145,146	82 - 39,140,143	83 - 144
84 - 47,141	85 - 25,42,147	86 - 86,43
87 - 45	88 - 25,46,148	89 - 26,47,149
92 - 148	93 - 52,147	94 - 149
95 - 32,117,133	96 - 33,118,134,150	97 - 34,119
98 - 32,120,135	99 - 33,121,136,151,152	100 - 34,122
101 - 137	102 - 123,150,151	103 - 138,152
104 - 38,124,139,153	105 - 39,125,140	106 - 126,141
107 - 38,127,142,154	108 - 39,128,143	109 - 144
110 - 145,154	111 - 129,146,153	112 - 130,147
113 - 53,131	114 - 148	115 - 53,132
116 - 149	117 - 57,95	118 - 58,96,155
119 - 59	120 - 57,98	121 - 58,99,156
122 - 59	123 - 155,156	124 - 62,104,157
125 - 63,105	126 - 106	127 - 62,107
128 - 63,108	129 - 111,157	130 - 112
131 - 73	132 - 73	133 - 76,95
134 - 77,96	135 - 76,98	136 - 77,99,158
138 - 158	139 - 81,104,159	140 - 82,105
141 - 106	142 - 81,107,160	143 - 82,108
145 - 160	146 - 111,159	150 - 102,155

continued on next page ...

... continued from previous page		
State - $\psi_2(x)$	State - $\psi_2(x)$	State - $\psi_2(x)$
151 - 102,156	152 - 158	153 - 157,159
154 - 160	155 - 123	156 - 123
157 - 153	159 - 153	

Table 3.6: Table of $\psi_2(x)$

3.8.3 Global Estimator

The global estimator computes the intersection of the two local estimates to obtain the global state estimate. Then, using the output-map (Table 3.4), the estimate of the condition of the system is obtained. Since timing information was used in the central diagnoser, it needs to be carried over to the distributed diagnoser also. Hence, the estimate thus obtained is stored. If the estimate remains the same until t_s, then the diagnoser declares a fault or otherwise, as the case may be. It should be noted that by earlier assumptions of permanent failure and availability of sufficient time for the system to settle, if the system stays in a fault state for t_s, the system will stay in that state for ever, i.e, the fault must have occurred. Table 3.7 lists the set of states and the corresponding set of states $\psi(x)$ (Section 2.4.2). Note that the table only lists the states for which $\psi(x)$ is nonempty.

State - $\psi(x)$	State - $\psi(x)$	State - $\psi(x)$	State - $\psi(x)$
0 - 9,10,11,12	1 - 19,20	3 - 34,35	4 - 40,41
5 - 44,45	6 - 48,49,50	7 - 51,52	8 - 53,54
13 - 59	14 - 64	15 - 67,68	16 - 71
17 - 72	18 - 73	23 - 87	24 - 90,91
30 - 97	31 - 100,101	33 - 102,103	36 - 106
37 - 109	38 - 110,111	42 - 112	43 - 113
46 - 114	47 - 115,116	55 - 119	56 - 122
58 - 123	60 - 126	62 - 129	65 - 130
66 - 131	67 - 13	70 - 132	75 - 137
			continued on next page ...

... continued from previous page			
State - $\psi(x)$	State - $\psi(x)$	State - $\psi(x)$	State - $\psi(x)$
79 - 141	80 - 144	85 - 147	88 - 148
89 - 149	96 - 150	99 - 151,152	104 - 153
107 - 154	118 - 115	121 - 156	124 - 157
136 - 158	139 - 159	142 - 160	

Table 3.7: Table of $\psi(x)$

Example 3.1 Consider that the system follows a path that takes it through the following sequence of states : 0, 1, 19, 59, 123. From Table 3.3 it is clear that Y follows the sequence (e,e,e,e), (l,e,e,e), (l,e,l,e) and (l,e,l,g) for the above sequence of states. For the sake of the diagnoser, assume that the system can start at any $x \in X$, i.e, $z_0 = X$. Therefore, $\tilde{z}_1(0) = \tilde{z}_2(0) = \{(x,x) \mid x \in X\}$. This implies that $\overline{z}_1(0) = \overline{z}_2(0) = X = \overline{z}(0)$.

Now, the output is (e,e,e,e). Let y_1 and y_2 be the outputs observed at local sites 1 and 2 respectively. Then, $y_1(1) = (e,e)$ and $y_2(1) = (e,e)$; hence,

$$\begin{aligned}
\tilde{z}_1(1) &= \left\{\begin{array}{l} (0,0),(3,3),(4,4),(7,7),(8,8),(9,9),(10,10),(11,11),(12,12),(32,32),\\ (33,33),(34,34),(35,35),(38,38),(39,39),(40,40),(41,41),(51,51),\\ (52,52),(53,53),(54,54),(102,102),(103,103),(110,110),(111,111) \end{array}\right\}, \text{ and}\\
\tilde{z}_2(1) &= \left\{\begin{array}{l} (0,0),(1,1),(2,2),(5,5),(6,6),(9,9),(10,10),(11,11),(12,12),(15,15),(16,16),\\ (19,19),(20,20),(23,23),(24,24),(27,27),(28,28),(29,29),(44,44),(45,45),\\ (48,48),(49,49),(50,50),(67,67),(68,68),(71,71),(87,87),(90,90),(91,91) \end{array}\right\}
\end{aligned}$$

From this we obtain $\overline{z}_1(1)$ and $\overline{z}_2(1)$ as follows:

$$\begin{aligned}
\overline{z}_1(1) &= \{0,3,4,7,8,9,10,11,12,32,33,34,35,38,39,40,41,51,52,53,54,102,103,110,111\}, \text{ and}\\
\overline{z}_2(1) &= \{0,1,2,5,6,9,10,11,12,15,16,19,20,23,24,27,28,29,44,45,48,49,50,67,68,71,87,90,91\}\\
\Rightarrow\ \overline{z}(1) &= \{0,9,10,11,12\}
\end{aligned}$$

Now, y_1 becomes (l,e), while y_2 stays the same. Hence,

$$\begin{aligned}
\tilde{z}_1(2) &= \left\{ \begin{array}{l} (0,1),(3,13),(4,14),(7,17),(8,18),(10,19),(12,20),(32,57),(33,58), \\ (34,59),(38,62),(39,63),(41,64),(52,72),(53,73),(102,123),(111,129) \end{array} \right\}, \text{ and} \\
\tilde{z}_2(2) &= \tilde{z}_2(1) \\
\Rightarrow\ \overline{z}_1(2) &= \{1,19,20\}, \text{ and} \\
\overline{z}_2(2) &= \psi_2(\overline{z}_2(1)) \ \supset\ \{1,19,20\} \\
\text{Therefore, } \overline{z}(2) &= \{1,19,20\}
\end{aligned}$$

Remark 3.1 *Note that $\overline{z}_2(2)$ is evaluated here only as a set containing $\{1,19,20\}$ and not completely, because this information is sufficient to evaluate $\overline{z}(2) = \cap_{i=1,2}\overline{z}_i(2)$.*

Now the event 812 (Valve 1 getting stuck closed) produces the transition from state 1 to state 19 and this transition does NOT produce any change in the output Y of the system. Hence $\overline{z}$ remains the same. Next, the system moves to state 59, producing an output (l,e,l,e); hence,

$$\begin{aligned}
\tilde{z}_1(3) &= \tilde{z}_1(2), \text{ and} \\
\tilde{z}_2(3) &= \left\{ \begin{array}{l} (0,3),(1,13),(2,21),(5,30),(6,31),(10,34),(11,35),(15,55),(16,56),(19,59), \\ (23,74),(24,75),(28,78),(44,97),(49,100),(50,101),(67,119),(71,122),(91,137) \end{array} \right\} \\
\Rightarrow\ \overline{z}_1(3) &= \psi_1(\overline{z}_1(2)) = \{13,14,17,18,19,20,59,64,72,73\}, \text{ and} \\
\overline{z}_2(3) &= \{13,59\} \\
\text{Therefore, } \overline{z}(3) &= \{13,59\}
\end{aligned}$$

Finally, the system reaches the state 123 with output (l,e,l,g) and

$$\begin{aligned}
\tilde{z}_1(4) &= \tilde{z}_1(3), \text{ and} \\
\tilde{z}_2(4) &= \left\{ \begin{array}{l} (3,33),(13,58),(21,77),(30,96),(31,99),(34,102),(35,103), \\ (55,118),(56,121),(59,123),(74,134),(75,136),(78,138), \\ (97,150),(100,151),(101,152),(119,155),(122,156),(137,158) \end{array} \right\} \\
\Rightarrow\ \overline{z}_2(4) &= \{58,123\}, \text{ and} \\
\overline{z}_1(4) &= \psi_1(\overline{z}_1(3)) \ \supset\ \{58,123\} \\
\text{Therefore, } \overline{z}(4) &= \cap_{i=1,2}\overline{z}_i(4) \ =\ \{58,123\}
\end{aligned}$$

Though the actual system is in state 123, the estimate of the diagnoser is that the system is either in state 123 or in state 58; the estimate of the condition being *V1sc* and *NORMAL* respectively. However, since a failure event (812) has occurred, the system stays in the same state even at the end of t_s seconds. Hence, at the end of t_s seconds, the diagnoser deduces correctly that the condition of the system is *V1sc*, i.e., Valve 1 is stuck closed. In fact, this can also be 'obtained' directly from the system by forcing the transition from 58 to 123 at the end of time t_s. Note that the transition from 58 to 123 by an 812 (event indicating that Valve 1 got stuck closed) exists already (Table 3.2). In addition, $\psi(58) = 123$ (from Table 3.7). $\square$

The distributed diagnoser so obtained in Section 3.8 was implemented as MATLAB code and tested out extensively to confirm that it provides the same diagnosis as the centralized diagnoser. The example listed above illustrates one such test case. Clearly, the decentralized diagnoser we obtained is superior to the centralized diagnoser; for, the decentralized diagnoser produces the same estimates as the central diagnoser but has the advantage of being decentralized and hence more fault-tolerant.

Chapter 4

Conclusion

This thesis describes a top-down procedure to construct a distributed diagnoser. The diagnoser so constructed produces exactly the same estimate as the centralized diagnoser of [HZ99] and hence the estimates of the condition of the system are identical. The major advantage of this approach is that the state and the condition of the system need not be known at the time of starting the diagnoser. This is to say that we have the freedom to initialize the diagnoser at any time during the operation of the system and not necessarily when the system is started. This feature makes this approach for diagnosis an excellent option for systems which require monitoring for failures after fault accommodation occurs; wherein the fault accomodation involves changes in the system parameters and the diagnoser along with it. We emphasize also that the decentralized architecture of the diagnoser presented in this thesis should make it more fault-tolerant than its centralized counterpart. Another important point to be noted here is that the distributed diagnoser is constructed from the centralized diagnoser without making any additional assumptions. Hence, for any system that has a centralized diagnoser, one can build a decentralized diagnoser without any loss of diagnosability whatsoever, i.e, the distributed diagnoser achieves the performance upper bound established by the centralized diagnoser of [HZ99].

We designed a discrete-time (DES) distributed diagnoser for a practical nonlinear continuous-time system. We also saw how the level of disturbance resulting from noise in the system could be used to define thresholds for identifying occurrence of failures. Furthermore, we also saw how incorporating suitable timing information into the system actually helps to detect the failure.

The two shortcomings of the approach are that it requires large memory and lots of communication. The memory shortcoming is largely offset by the fact that memory is in fact very cheap nowadays. However, the burden on the communication channel needs to be offset. In fact, Example 3.1 highlights an important fact that it might be sufficient to compute or communicate only a certain minimum set of states, but with an additional feedback from the local state estimators to the corresponding local observers. Methods to reduce the complexity of the system and communication present an interesting research topic. Also, a method to obtain the distributed diagnoser directly from a decentralized plant (a bottom-up approach) would largely simplify the process of obtaining the distributed diagnoser and warrants research.

Appendix A

Proof of Theorem 2.1

Theorem: Assume the single-failure scenario and $z_0 = X$. A permanent failure F_i is diagnosable if and only if:

1. From every $x \in X_{F_i}$, there is at least one transition to another state in X_{F_i} unless $\lambda(x)$ is F_i-indicative,

2. there is no cycle in X_{F_i} consisting of states having the same output unless the output symbol is F_i-indicative, and

3. there are no F_i indeterminate cycles in the diagnoser. □

Proof: (reproduced from [HZ99])

Lemma A.1 *(Lemma 2.2 in [HZ99]) Consider a path* $z_0 \stackrel{y_1}{\to} z_1 \stackrel{y_2}{\to} z_2 \ldots \stackrel{y_n}{\to} z_n$ $(n \geq 1)$ *in the diagnoser. For any* $x_n \in z_n$*, there exist* $x_i \in z_i$ $(1 \leq i \leq n-1)$ *such that* $x_i \Rightarrow x_{i+1}$ *for* $1 \leq i \leq n-1$.
□

Conditions 1 and 2 guarantee that after F_i occurs, new output symbol(s) will be generated and the output sequence can only terminate (if it does) in an F_i-indicative symbol. Suppose either 1 or 2 does not hold. After F_i occurs, the output sequence may end in a non-F_i-indicative output. If the diagnoser is initialized after this last symbol is generated, then it will not be able to detect and isolate the failure. Therefore, both 1 and 2 are necessary for diagnosability; from now on, assume they hold.

After the occurrence of F_i and the generation of a new output symbol, the diagnoser state will be either F_i-certain or F_i-uncertain. If it is F_i-certain, then it will remain F_i-certain (because F_i is permanent) and F_i is diagnosed. If the diagnoser state is F_i-uncertain, then by condition 3 and the fact that the number of F_i-uncertain states is bounded, after the generation of a bounded number of output symbols, the diagnoser will reach an F_i-certain state (In general, the diagnoser gets trapped indefinitely in a cycle of F_i uncertain states only if the cycle is F_i-indeterminate). Let n_i denote the number of events that it takes the diagnoser to detect and isolate F_i. By condition 3, after the occurrence of the failure, the diagnoser can visit an F_i-uncertain state z at most $|z \cap X_{F_i}|$ times. Hence,

$$n_i \leq c_i \times p_i + p_i$$

where $c_i = \Sigma\{|z_k \cap X_{F_i}| \mid z_k \in Z \& z_k \text{ is } F_i\text{−uncertain}\}$ and p_i is the length of the longest path of faulty states having the same non-F_i-indicative output. By condition 2, $p_i \leq |X_{F_i}|$. Also $c_i \leq |X_{F_i}| \cdot |Z|$. Therefore,

$$n_i \leq |X_{F_i}|(|X_{F_i}| \cdot |Z| + 1)$$

As a result, F_i is diagnosable with $N_i = |X_{F_i}|(|X_{F_i}| \cdot |Z| + 1)$.

Conversely, if condition 3 is not true, then there exists an output sequence $y_1, y_2, \ldots, y_n$ that can take the diagnoser into a state z_k belonging to an F_i-indeterminate cycle. Let $x_n \in z_n$ belong to an underlying faulty cycle in the RTS. By Lemma A.1, there exist states $x_1, \ldots, x_{n-1}$ such that $x_i \Rightarrow x_{i+1}$ for $1 \leq i \leq n-1$. After reaching x_n, the RTS state may remain on the underlying fault cycle causing the diagnoser to stay on the F_i-indeterminate cycle indefinitely. Therefore, there exists a trajectory for the system leading to states in X_{F_i} such that the corresponding output sequence throws the diagnoser into a cycle of F_i-uncertain states and keeps it there indefinitely. Hence, F_i is not diagnosable. $\square$

Appendix B

MATLAB Code

bas.m

```
% bas.m
% two eng model
% used for non-linear simulation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** initialization ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
basIni;
inputIni;
diagnoserIni;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** simulation parameters ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dt  =  0.005;
TFINAL  =  85;
T = 0:dt:TFINAL;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** simulation ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%

for t = dt:dt:TFINAL
   p1  = input_p1(floor(t/dt));
   p2  = input_p2(floor(t/dt));
   p10 = input_p10(floor(t/dt));

   limb1;
   limb2;
   limb3;

   limb1new;
   limb2new;
   limb3new;

   limb1_2plot;
   limb2_2plot;
   limb3_2plot;

   fault = 0;

   if (floor(t) == t & floor(t/15) == t/15)
        distribDiag;
        %centralDiag;
        fault_2plot = [fault_2plot; fault];
        ct_2plot = [ct_2plot; t];
   end

   if (fault ~= 0)
        flowBalance = 0;
```

```
   end
end

t_2plot = T;
clear T;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** plot results ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plotSensed;
plotInput;
plotFault;
```

basIni.m

```
% ***** basIni.m
% ***** initializes the BAS

clear;
close all hidden;

gamma   =   1.4;
g       =   384;
R       =   639.6;

v1      =   3600;
v2      =   10000;
v8      =   12000;
v9      =   2000;
v10     =   10000 * 1728;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** limb1 constants  ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
k1      =   0.5;
tau11   =   .02;
tau12   =   .01;
d1      =   4;
t1      =   910;
c1112   =   3;
c125    =   4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** limb1 initial values  ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p11sp   =   60;
p11     =   64;
beta1   =   0;
x1      =   0;
p12     =   53;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** limb2 constants  ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k2      =   0.5;
tau21   =   .02;
tau22   =   .01;
d2      =   4;
t2      =   910;
c2122   =   3;
k225    =   0.4;
c225    =   4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** limb2 initial values  ***** %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p21sp   =   60;
p21     =   58;
beta2   =   0;
x2      =   0;
p22     =   55;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** limb3 constants  ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
v12     =   2000;
k5      =   0.5;
tau51   =   0.02;
tau52   =   0.01;
d5      =   4;
c5110   =   5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** limb3 initial values  ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p5      =   49;
p51sp   =   25;
beta5   =   0;
x5      =   0;
p51     =   20;

%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** controller ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%
err1w   =       0;      err1p   =       0;
err2w   =       0;      err2p   =       0;
err5w   =       0;      err5p   =       0;
```

```
xc1w    =       0;      xc1p    =       0;
xc2w    =       0;      xc2p    =       0;
xc5w    =       0;      xc5p    =       0;

kp1w    =       7;      kp1p    =       0.5;
ki1w    =       2;      ki1p    =       1;


kp2w    =       7;      kp2p    =       0.5;
ki2w    =       2;      ki2p    =       1;

kp5w    =       4;      kp5p    =       0;
ki5w    =       1;      ki5p    =       0;

%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** setpoints ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%
w12sp   =       1;      p11ref  =       p11sp;
w22sp   =       1;      p21ref  =       p21sp;
w5sp    =       2;      p51ref  =       p51sp;

dp11sp  =       0;
dp21sp  =       0;
dp51sp  =       0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** flow-balance ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
flowBalance     =       1;

%%%%%%%%%%%%%%%%%%%%%%%
```

```
% ***** sensors ***** %
%%%%%%%%%%%%%%%%%%%%%%%
w12m    =       0;
w22m    =       0;
w5m     =       0;

p11m    =       0;
p21m    =       0;
p51m    =       0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** values to plot ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t_2plot     =   0;

w11_2plot   =   0;
w21_2plot   =   0;
w12_2plot   =   0;
w22_2plot   =   0;
w13_2plot   =   0;
w24_2plot   =   0;
w51_2plot   =   0;
w5_2plot    =   0;

p11_2plot   =   p11;
p12_2plot   =   p12;
p21_2plot   =   p21;
p22_2plot   =   p22;
p5_2plot    =   p5;
p51_2plot   =   p51;

beta1_2plot =   beta1;
```

```
beta2_2plot =   beta2;
beta5_2plot =   beta5;
```

inputIni.m

```
% inputIni.tex
% initializes the input

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** normal input ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

input_p1 = 120 * ones(size(T));
input_p2 = 120 * ones(size(T));
input_p10 = 15 * ones(size(T));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** modified input ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%-- uncomment modifications needed

%input_p1  = random('norm',120,3.24, size(T));    % std = 1.5 /100 * 120
%input_p2  = random('norm',120,3.24, size(T));    % std = 1.5 /100 * 120
%input_p10 = random('norm',15,.050625, size(T));  % std = 1.5 /100 * 15
```

diagnoserIni.m

```
% diagnoserIni.m
% initializes the diagnoser parameters

fault_2plot = 0;
ct_2plot = 0;
```

limb1.m

```
% limb1.m
% simulation code for limb1

a1      =   pi * d1 * d1 / 4 * (1 - cos(beta1 * pi/180));
phi     =   max(p11/p1, 0.5283);
w11     =   a1*p1*sqrt(2*gamma*g/(R*t1*(gamma-1))) * sqrt(phi^(2/gamma) - phi^((1+gamma)/gamma));

if(imag(w11)~=0)    %%%% to detect pgm error--begins
et = t
        ew11 = w11
        ep11 = p11
        break;
end                 %%%% to detect error--ends

t12     =   t1;
w12     =   1/c1112 * (p11-p12);
w13     =   1/c125 * (p12-p5);
dx1     =   k1*(p11sp-p11)*dt - (x1 + k1*tau11*(p11sp-p11))/tau12 * dt ;
dp11    =   gamma * R * t12 * (w11 - w12) / v8 * dt;
dp12    =   gamma * R * t12 * (w12 - w13) / v1 * dt;

%%%%%%%%%%%%%%%%%%%%%%%
% ***** sensor *****  %
%%%%%%%%%%%%%%%%%%%%%%%
w12m    =   w12m + (w12 - w12m) * dt/.5;
p11m    =   p11m + (p11 - p11m) * dt/.1;

%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** controller ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%
err1w   =  flowBalance * (w12sp - w12m);
xc1w    =  xc1w + ki1w * err1w * dt;
```

```
err1p   =  p11ref-p11sp;
xc1p    =  xc1p + ki1p * err1p * dt;

dp11sp  =  kp1w * err1w + xc1w + kp1p * err1p + xc1p;
```

limb2.m

```
% limb2.m
% simulation code for limb2

a2      =   pi * d2 * d2 / 4 * (1 - cos(beta2 * pi/180));
phi     =   max(p21/p2, 0.5283);
w21     =   a2*p2*sqrt(2*gamma*g/(R*t2*(gamma-1))) * sqrt(phi^(2/gamma) - phi^((1+gamma)/gamma));
if(imag(w21)~=0)    %%%%% to detect error begins
et = t
        ew21 = w21
        ep21 = p21
        break;
end                 %%%%% to detect error ends

%t22     =   t2 * (p21/p2) ^ (1 - 1/gamma);
t22     =   t2;
w22     =   1/c2122 * (p21-p22);
w24     =   1/c225 * (p22-p5);
dx2     =   k2*(p21sp-p21)*dt - (x2 + k2*tau21*(p21sp-p21))/tau22 * dt ;
dp21    =   gamma * R * t22 * (w21 - w22) / v8 * dt;
dp22    =   gamma * R * t22 * (w22 - w24) / v2 * dt;

%%%%%%%%%%%%%%%%%%%%%%%
% ***** sensor *****  %
%%%%%%%%%%%%%%%%%%%%%%%
w22m    =   w22m + (w22 - w22m) * dt/.5;
p21m    =   p21m + (p21 - p21m) * dt/.1;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** controller ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%
err2w   =  flowBalance * (w22sp - w22m);
xc2w    =  xc2w + ki2w * err2w * dt;

err2p   =  p21ref-p21sp;
xc2p    =  xc2p + ki2p * err2p * dt;

dp21sp  =  kp2w * err2w + xc2w + kp2p * err2p + xc2p;
```

limb3.m

```
% limb3.m
% simulation code for limb3

if((w13+w24)~=0)
    t5      =   (w13 * (t12-460) + w24 * (t22-460))/(w13 + w24) + 460;
else
    t5      =   746;
end
%t51     =   t5 * (p51/p5) ^ (1 - 1/gamma);
t51     =   t5;
%t7      =   t51 * (p10/p51) ^ (1 - 1/gamma);
t7      =   t5;
a5      =   pi * d5 * d5 / 4 * (1 - cos(beta5 * pi/180));
phi     =   max(p51/p5, 0.5283);
w51     =   a5*p5*sqrt(2*gamma*g/(R*t5*(gamma-1))) * sqrt(phi^(2/gamma) - phi^((1+gamma)/gamma));
if(imag(w51)~=0)    %%%%% to detect error begins
        et = t
        ew51 = w51
        ep51 = p51
        ep5 = p5
```

```
        break;
end                 %%%%% to detect error ends
w5      =   1/c5110 * (p51-p10);
t9      =   (w5 * t5 + w6 * t6) / (w5 + w6);
dx5     =   k5*(p51sp-p51)*dt - (x5 + k5*tau51*(p51sp - p51))/tau52 * dt;
dp5     =   gamma * R * t5 * (w13 + w24 - w51) / v12 * dt;
dp51    =   gamma * R * t51 * (w51 - w5) / v9 * dt;

%%%%%%%%%%%%%%%%%%%%%%%
% ***** sensor *****  %
%%%%%%%%%%%%%%%%%%%%%%%
w5m     =   w5m + (w5 - w5m) * dt/.5;
p51m    =   p51m + (p51 - p51m) * dt/.1;

%%%%%%%%%%%%%%%%%%%%%%%%%%
% ***** controller ***** %
%%%%%%%%%%%%%%%%%%%%%%%%%%
err5w   =  flowBalance * (w5sp - w5m);
xc5w    =  xc5w + ki5w * err5w * dt;

err5p   =  p51ref-p51sp;
xc5p    =  xc5p + ki5p * err5p * dt;

dp51sp  =  kp5w * err5w + xc5w + kp5p * err5p + xc5p;
```

limb1new.m

```
% limb1new.m
%

x1      =   x1 + dx1;
alpha1  =   (x1 + k1 * tau11 * (p11sp-p11))/tau12;
beta1   =   beta1 + alpha1 * dt;
%if (t>35)
```

```
% beta1   =   0;       %%%%%%%% valve 1 is sc -- 12
%end
beta1   =   min(90, max(0, beta1));
p11     =   min(p1, p11 + dp11);
p12     =   p12 + dp12;

%%%%%%%%%%%%%%
% controller %
%%%%%%%%%%%%%%
p11sp   =   p11ref + dp11sp;
```

limb2new.m

```
% limb2new.m
%

x2      =   x2 + dx2;
alpha2  =   (x2 + k2 * tau21 * (p21sp-p21))/tau22;
beta2   =   beta2 + alpha2 * dt;
if (t>25)
beta2   =   0;    %%%%%% valve 2 is sc. -- 22
end
beta2   =   min(90, max(0, beta2));
p21     =   min(p2, p21 + dp21);
p22     =   p22 + dp22;

%%%%%%%%%%%%%%
% controller %
%%%%%%%%%%%%%%
p21sp   =   p21ref + dp21sp;
```

limb3new.m

```
% limb5new.m
%
```

```
x5      =   x5 + dx5;
alpha5  =   (x5 + k5 * tau51 * (p51sp-p51))/tau52;
beta5   =   beta5 + alpha5 * dt ;
%if (t>35)
% beta5   =   90;    %%%%%%%%%% valve 5 is so. -- 50
%end
beta5   =   min(90, max(0, beta5));
p5      =   max(p51, p5 + dp5);
p51     =   min(p5, p51 + dp51);

%%%%%%%%%%%%%%
% controller %
%%%%%%%%%%%%%%
p51sp   =   p51ref + dp51sp;
```

limb1_2plot.m

```
% limb1_2plot

p11_2plot   =   [p11_2plot; p11];
p12_2plot   =   [p12_2plot; p12];
w11_2plot   =   [w11_2plot; w11];
w12_2plot   =   [w12_2plot; w12];
w13_2plot   =   [w13_2plot; w13];
beta1_2plot =   [beta1_2plot; beta1];
```

limb2_2plot.m

```
% limb2_2plot

p21_2plot   =   [p21_2plot; p21];
p22_2plot   =   [p22_2plot; p22];
w21_2plot   =   [w21_2plot; w21];
w22_2plot   =   [w22_2plot; w22];
```

```
w24_2plot   =   [w24_2plot; w24];
beta2_2plot =   [beta2_2plot; beta2];
```

limb3_2plot.m

```
% limb5_2plot
%

p5_2plot    =   [p5_2plot; p5];
p51_2plot   =   [p51_2plot; p51];
w51_2plot   =   [w51_2plot; w51];
w5_2plot    =   [w5_2plot; w5];
beta5_2plot =   [beta5_2plot; beta5];
```

centralDiag.m

```
% centralDiag.m
% centralize diagnoser for BAS

if (p11m - p11ref < -0.5)
        if (p21m - p21ref < -0.5)
            if (p51m - p51ref > 0.5)
                if (w12m - w12sp > 0.1 & w22m - w22sp > 0.1 & w5m - w5sp > 0.2)
    %%% valve 5 is stuck open %%%
                      fault = 50
                      ct = t
        else
    %%% diagnoser failure %%%
                end
            elseif (w12m - w12sp < -0.1 & w22m - w22sp > 0.1)
   %%% valve 1 is stuck closed %%%
                     fault = 12
                     ct = t
            elseif (w12m - w12sp > 0.1 & w22m - w22sp < -0.1)
   %%% valve 2 is stuck closed %%%
```

```
                     fault = 22
                     ct = t
    else
   %%% diagnoser failure %%%
            end
else
    %%% diagnoser failure %%%
        end
elseif (p11m - p11ref > 0.5)
        if (p21m - p21ref > 0.5)
                if (p51m - p51ref < -0.5)
                   if (w12m - w12sp < -0.1 & w22m - w22sp < -0.1 & w5m - w5sp < -0.2)
      %%% valve 5 is stuck closed %%%
                      fault = 52
                      ct = t
   else
      %%% diagnoser failure %%%
                   end
                elseif (w12m - w12sp > 0.1 & w22m - w22sp < -0.1)
       %%% valve 1 is stuck open %%%
                       fault = 10
                       ct = t
                elseif (w12m - w12sp < -0.1 & w22m - w22sp > 0.1)
       %%% valve 2 is stuck open %%%
                       fault = 20
                       ct = t
else
       %%% diagnoser failure %%%
                end
else
%%% diagnoser failure %%%
        end
```

```
else
%%% diagnoser failure %%%
end

fault_2plot = [fault_2plot; fault];
```

plotSensed.m

```
% plotSensed.m
% to plot the measured(sensed) variables

figure;
subplot(3,2,1);
    plot(t_2plot, p11_2plot);
    title('p11');

subplot(3,2,2);
    plot(t_2plot, w12_2plot);
    title('w12');

subplot(3,2,3);
    plot(t_2plot, p21_2plot);
    title('p21');

subplot(3,2,4);
    plot(t_2plot, w22_2plot);
    title('w22');

subplot(3,2,5);
    plot(t_2plot, p51_2plot);
    title('p51');

subplot(3,2,6);
    plot(t_2plot, w5_2plot);
```

```
    title('w5');
```

plotInput.m

```
% plotInput.m
% two eng model

figure;
hold on;
subplot(3,1,1);
    plot(t_2plot, input_p1);
    title('p1');

subplot(3,1,2);
    plot(t_2plot, input_p2);
    title('p2');

subplot(3,1,3);
    plot(t_2plot, input_p10);
    title('p10');
```

plotFault.m

```
% plotFault.m
% to plot the fault

figure;
plot(ct_2plot,fault_2plot);
```

Bibliography

[Buc64] P. S. Buckley. *Techniques of Process Control.* New York: John Wiley, 1964.

[CP99] J. Chen and Ron J. Patton. *Robust model-based fault diagnosis for dynamic systems.* Kluwer Academic Publishers, Boston, USA, 1999.

[DLT00] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Journal of Discrete Event Dynamical Systems: Theory and Application*, 10(1-2):22–86, January 2000.

[Fra90] P. Frank. Fault diagnosis in dynamic systems using analytical and knowledge based redundancy. *Automatica*, 26:459–474, 1990.

[Goo97] Robert B. Goodman. *A Primer on Pneumatic Valves and Controls.* Krieger Publishing Company, 1997.

[HCdK92] Hamscher, Console, and de Kleer. *Readings in Model-Based Diagnosis.* Morgan Kaufmann Publishers, 1992.

[Hon01] Honeywell. Honeywell ASCa Inc. handout, 2001.

[HZ99] Shahin Hashtrudi Zad. *Fault Diagnosis in Discrete-Event and Hybrid Systems.* PhD thesis, Dept. of Electrical and Computer Engg., University of Toronto, August 1999.

[Joh00] Curtis D. Johnson. *Process Control Instrumentation Technology.* Prentice Hall, 2000.

[Kar01] Johan Karlsson. Diagnosis of the air distribution system of the JAS39 Gripen environmental control system. Master's thesis, Vehicular systems, Linköping Institute of Technology, February 2001.

[Lev95] N.G. Leveson. *Safeware: System Safety and Computers.* Addison- Wesley, Mass., 1995.

[LGTL85] W. S. Lee, D.L. Grosh, F.A. Tillman, and C.H. Lie. Fault tree analysis, methods and applications - a review. *IEEE Trans. on Reliability*, 34(3):194–203, 1985.

[Sha53] Ascher H. Shapiro. *The Dynamics and Thermodynamics of Compressible Fluid Flow.* The Ronald Press, 1953.

[Shi98] F. G. Shinskey. *Process Control Systems.* McGraw Hill, 1998.

[SSL+95] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. Automat. Contr.*, 40(9):1555–1575, September 1995.

[SSL+96] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Trans. Automat. Contr. Syst. Tech.*, 4(2):105–124, March 1996.

[Su00] Rong Su. Decentralized fault diagnosis for discrete-event systems. Master's thesis, Dept. of Electrical and Computer Engg., University of Toronto, April 2000.

[SW87] W.T. Scherer and C.C. White. *Fault Detection and Reliability: Knowledge Based and Other Approaches.* Pergamon Press, 1987.

[Won02] W.M. Wonham. *Notes on Control of Discrete-Event Systems.* System Control Group, University of Toronto, July 2002. http://www.control.utoronto.ca/people/profs/wonham/wonham.html.