INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

REAL-TIME MONITORING AND DIAGNOSIS IN DYNAMIC SYSTEMS USING PARTICLE FILTERING METHODS

BY

RUBEN MORALES-MENENDEZ

A DISSERTATION
SUBMITTED TO THE GRADUATE PROGRAM IN ELECTRONICS,
COMPUTING, INFORMATION AND COMMUNICATION
AND THE COMMITTEE OF GRADUATE STUDIES OF
INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES
DE MONTERREY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN INTELLIGENT SYSTEMS

MONTERREY, NUEVO LEON, MEXICO, MAY 2003

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS MONTERREY

REAL-TIME MONITORING AND DIAGNOSIS IN DYNAMIC SYSTEMS USING PARTICLE FILTERING METHODS

BY

RUBÉN MORALES-MENÉNDEZ

A DISSERTATION
SUBMITTED TO THE GRADUATE PROGRAM IN ELECTRONICS, COMPUTING, INFORMATION AND COMMUNICATION
AND THE COMMITTEE OF GRADUATE STUDIES OF
INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN INTELLIGENT SYSTEMS

MONTERREY, NUEVO LEÓN, MÉXICO , MAY 2003

©Copyright by Rubén Morales-Menéndez, 2003
All Rights reserved

External Advisors Acknowledgements

David Poole

I would like to thank David for several reasons. Firstly, he allowed me to continue my doctoral studies as a visiting scholar in the Department of Computer Science at the University of British Columbia for three years. Secondly, as my advisor, he provided invaluable support, guidance and insight which helped me accomplish my research effectively. Thirdly, during this time, he gave me the opportunity to attend many interesting and inspiring lectures, seminars and reading groups facilitated by world-renowned researchers in many fields; this helped me to complete the theoretical component of my doctoral program.

Due to the generosity of the Computer Science Department at UBC, I had access to many vital resources such as current related journals and publications, software, and the mobile robot which played an important role in part of my research.

Nando de Freitas

Nando provided support during my last 2 years at UBC and worked with David to establish the state-of-the-art Monte Carlo Particle Filtering method as the focus of my research. I would like to thank Nando for exposing me to this method so that I could complete my work. He patiently enlightened me on the software, research and current scientific thought of this new field. His seminar and Probabilistic Machine Learning course were crucial to my education. His ideas and guidance allowed me to complete my research. Nando (and David) spent much time reviewing my work and guiding me in the development of two papers. These papers were accepted at NIPS[1] 2002 and ACC[2] 2003. Another journal article[3] has also been submitted thanks to Nando's efforts in coordinating contact with the people at NASA.

Finally, I would also like to thank David and Nando for all they taught me about how to do meaningful research, generate new ideas, create/develop research groups and finalize a project effectively. These details are the most valuable knowledge I gained.

Thanks David and Nando for investing time in me during these years.

[1] Neural Information Processing Systems

[2] American Control Conference

[3] Proceedings IEEE, Special Issue on Sequential State Estimation

Acknowledgements

I would like to thank Francisco Cantú for introducing me to working in this field. He granted me the freedom to explore various frameworks while trying to define my research project, and he helped me secure a position as a visiting scholar at the University of British Columbia for three years. Most importantly, Francisco always trusted in me during my most difficult struggles. His patience, encouragement and suggestions were very crucial in the completion of my academic career.

I would also like to thank my committee members Luis Enrique Sucar, Arturo Nolazco and Ricardo Ramirez for taking the time to review my thesis. Their unconditional support was very important. Specially, Arturo helped with suggestions for the presentation and Luis Enrique provided many detailed recommendations and ideas for improving my thesis.

As an ITESM professor, I am very thankful for the exceptional support that I received from many people, but especially from Alberto Bustani, Carlos Narváez, Bertha Dávila, Eugenio Garcia, and Fernando Jaimes. They enabled me to achieve this goal.

From the ITESM side, the doctoral program coordinators Rogelio Soto and Hugo Terashima also helped me on numerous occasions. I also need to thank Francisco Calleja for helping me with my experimental tests, Jorge Limón for his excellent discussions regarding state space representation, and Amparo Herrera for taking care of my affairs in Monterrey during these years.

I would like to thank Federico Guedea and Jim Mutch whom I feel fortunate to have met. During my first year at ITESM, Federico was an incredible colleague who supported me in my work as a professor and Ph.D. student; his support was invaluable. During the following years, he provided a lot of motivation and ideas that were essential to my research. Jim at UBC made it possible to implement my ideas in a mobile robot, and all of his feedback, suggestions and support were important for my research. Their contributions can be seen throughout my research.

Frank Hutter at the Darmstadt University of Technology gave me fresh ideas for optimizing my code and improving my conclusions. He also tested the inference algorithm with some planetary rovers at NASA, making me more confident of my results.

From the UBC side, thanks to Alan Mackworth, Uri Ascher, Raymond Ng, Cristina Conati, David Poole, Nando de Freitas, Paul Gustafson and Martin Puterman who allowed me to informally attend their impressive courses. Their knowledge and experience helped me to complete my research.

My thanks to the Government of Canada (International Council for Canadian Studies), who sponsored me for one year.

I would like to thank my family at home: *Juan José*, *Lety*, *Gil*, *Liz*, my *father*, and my in-laws whose love and support I could always count on. Thanks also to *Lupis* for her unconditional caring.

Finally, I would like to thank the most important people in my life, my wife and daughters. Thanks to *Pily*, *Montserrat*, *Georgy* and *Marianne* for giving me new dreams to pursue. I am always so proud of all my family. They were my only motivation and inspiration each and every day that allowed me to continue and finish this dissertation.

Thank *God* for my family and friends!

Dedication

To my *Mom*† and *Grandmother*†
who passed away after I had started my research,
but whose love lives in my heart.

To my *Grandfather*†,
my second father,
who always took care of me.

Abstract

Fault diagnosis is a critical task for many real-world processes. The diagnosis problem is that of estimating the most probable state of a process over time given noisy observations. For many applications, the complex dynamics of the process requires reasoning with both discrete and continuous variables, so a hybrid model such as the jump Markov linear Gaussian (*JMLG*) model is needed. The *JMLG* model has a set of discrete modes to represent the fault states and a set of continuous parameters for the continuous variables.

Although diagnosis is a simple procedure in principle, it is quite costly even for processes that are compactly represented, because the belief state is typically exponential in the number of state variables. Computing exact diagnosis is an intractable problem. Therefore, we must use numerical approximation methods such as *Particle Filtering* (*PF*).

PF is a state-of-the-art Markov chain Monte Carlo method which can diagnose dynamic systems by approximating the belief state as a set of *particles*. *PF* sequentially computes an approximation to the posterior probability distribution of the process states given the observations. This nonparametric approach has several advantages; it can approximate any probability distribution and consequently can be used to monitor systems with changing or uncertain structure.

Rao-Blackwellized Particle Filtering (*RBPF*) is a Particle Filtering variant that combines a *PF* for sampling the discrete modes with *Kalman Filters* for computing the distributions of the continuous states. This reduces the computational cost because the continuous states are represented by the sufficient statistics of the continuous distributions.

In this research, we show that it is possible to enhance the *RBPF* algorithm to sample the discrete modes directly from the posterior probability distribution. It is also possible to select the fittest particles before the sampling step. These improvements result in a more efficient algorithm, *look-ahead Rao-Blackwellized Particle Filtering* (*la-RBPF*). *La-RBPF* essentially performs one-step look-ahead to select good sampling regions. We show that the overhead of the extra processing per particle is more than compensated for by the decrease in diagnosis error and variance.

La-RBPF provides several additional advantages. It requires fewer particles to achieve the same approximation accuracy. Moreover, because the discrete modes are sampled from the posterior probability distribution, if a fault state appears, *la-RBPF* will detect the fault no matter how low its prior probability distribution.

Another additional result of this research is a learning method for the *JMLG* parameters. This modelling procedure combines the *Least Squares Estimation* and *Expectation-Maximization* algorithms.

The proposed algorithms were intensively tested in several real-world systems having industrial characteristics, and compared with the existing *PF* and *RBPF* algorithms. The results consistently demonstrated *la-RBPF*'s advantages.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter I give a very brief and simple introduction to this research. For ease of reading, citations, equations and justifications are kept to a minimum. Except for the motivation section, every following concept or idea is fully discussed or analyzed in a later chapter.

1.1 Motivation

System monitoring and timely fault detection capabilities are critical requirements of many modern systems. For years these features have only been of utmost importance in safety critical systems such as civil and military aviation, or nuclear power plants, etc. However, recently, other factors have been playing a major role in recognizing the need for these capabilities in other industrial systems. For the time being, by the term *fault* we consider failures, errors, malfunctions, deviation or disturbances in the functional process that can lead to undesirable or intolerable behaviour of the process.

It is no surprise that the general problem of fault detection and isolation has received considerable attention in the literature of reliability engineering, control engineering and computer science (the artificial intelligence community). There are many reasons that have made the automatic fault detection and isolation and accommodation problem to become an active area for research in a wide variety of industries and systems. Examples of contributing factors are:

- Processes in the chemical and petrochemical industrial are becoming larger and more complex. Associated of this trend imply that each hour of down time is more expensive, and that the source of malfunction or fault is more difficult to locate. As industrial processes enlarge, the total amount of energy and material being handled increases, making early and correct fault detection and diagnosis imperative both from the point of view of the system safety as well as reduced manufacturing costs. The purpose of monitoring for faults is to reduce the occurrence of sudden, disruptive, or dangerous outages, equipment damage, and personal accidents, and to assist in the operation of the maintenance program.

- The increased level of sophistication of many industrial and consumer goods due to the advances in electronics and computer technology, and the same time decrease in processor's costs. Today's automobiles are a good example of this trend. The auto manufactures have incorporated a huge amount of electronics in recent cars. Many functions such as anti-block brakes, security systems, traction control, temperature control, etc are performed automatically.

- Autonomous underwater vehicles (*AUV*)and remotely operated vehicles (*ROV*) have been receiving increasing attention over the last years due to their significant impact in several underwater operations, [Antonelli, 2003]. Examples are monitoring and maintenance of off-shore structures or pipelines, and exploration of the ocean floor. The benefit in the use of unmanned vehicles is in terms of safety, due to the possibility to avoid the risk of manned missions, and economic. Autonomous underwater vehicles are generally required to operate over long periods of time in unstructured environments in which an undetected failure usually implies loss of the vehicle. It is clear that, even in case of failure detection, in order to terminate the mission, or simply to recover the vehicle, a fault tolerant strategy, must be considered. In case of the use of remotely operated vehicles, a human operator is in charge of remotely operated vehicles, a failure detection strategy would help the human decision making process.

- The usefulness of autonomous systems such as robots in hazardous or dangerous situations is highly dependent on their reliability [Visinsky *et al.*, 1995]. Chemicals and radiation can damage robotic components, whereas actions of malfunctioning robots can make environments more hazardous. As humans usually cannot enter hazardous environments to repair or remove a failed robot, such failures can be very expensive.

- Modern military aircraft have some unique features compared to other industrial applications, and the design of control systems capable of compensating for a wide range of failures poses numerous challenges, [Boskŏivć and Mehra, 2002]. Modern military aircrafts are characterized by highly reliable components and flight software, however, critical subsystem or component failures can cause instability of the closed-loop control system and loss of the aircraft. While not every failure is critical, the fact that even a single failure may lead to catastrophic consequences makes the fault detection and isolation system design highly challenging.

- The environmental concern is now a new reason for fault diagnosis systems[1]. The California Air Resource Board, and Environmental Protection Agency legislations require as of 1998, that On Board Diagnostics II be rolled into all light duty vehicles sold in North American fleet. Basically, it requires fault detection capability for all vehicle components whose failure could result in emission levels beyond a certain threshold.

- In the last 10 years, three space missions have disappeared before reaching Mars, [Verma *et al.*, 2001]. *Mars Observer* was launched on Sept 25,1992 from the Kennedy Space Center aboard a Titan III rocket. On August 21, 1993 the spacecraft was lost in the vicinity of Mars after (most probably) an explosion of the fuel and oxidizer elements when the spacecraft began its manoeuvring sequence for Martian orbital insertion.*Mars Climate Orbiter* failed to achieve Mars orbit on Sept 23, 1999. A failure to recognize and correct an error in a transfer of information between the spacecraft team in Colorado and the mission navigation in California lead to the loss of the spacecraft, [NASA, 1999]. *Mars Polar Lander* and the two Deep Space 2 micro-probes were integrated on a common cruise stage for the trip. Separation of the micro-probes and the lander was planned to occur about 10 minutes prior to the Mars landings. The design of the lander precluded any communications from the period shortly before separation from the cruise stage until after Mars landing. The planned communications after landing did not occur, resulting in the conclusion that the *Mars Polar Lander* mission failed. Several possible failure causes are presented, which include loss of control due to spacecraft dynamic effects or fuel migration, local characteristics of the landing site beyond the capabilities of the lander, and the parachute covering the the lander after touchdown. The most probable cause of failure is that spurious signals gave a false indication that the lander had landed, resulting in a premature shutdown

[1] *Update, The Center for Systems Science, Simon Fraser University, Canada*

of the lander engines and the destruction of the lander when it crashed into the Mars surface on Dec 3, 1999, [NASA, 2000].

In the processes and systems that were described above, in order to have the efficient operation of the process and to increase the reliability and safety, prompt detection of faulty situations (*fault detection*) and the fast identification (*isolation*) of the most probable causes (*faults*) need to be addressed. Fault Detection and Isolation (*FDI*) can be carried out using analytical or functional information about the system being monitored, i.e. based on a mathematical model of the system. This approach is known as analytical redundancy, which is also known as model-based or quantitative *FDI*. Model-based is currently the subject of extensive research and is being used in highly reliable control systems due to the fact that analytical redundancy based techniques are economical and powerful. Fault detection and diagnosis can be applied at many stages:

- Detection of an incipient fault
- Early detection of faulty condition
- Real time determination of causes of a malfunction
- Prediction of a suitable course of action to take to rectify the abnormal condition
- Post-failure diagnosis of the cause of failure

process's requirements define the best place for the *FDI* system.

A key component in a *FDI* system is process monitoring, which is a continuous real-time task of recognizing anomalies in the behaviour of a dynamic system and identifying the underlying faults. This task has three important difficulties:

1. *Diagnosis must be on line*, process systems are designed for continuous operation and are capable of operating with multiple minor faults; shutdown for diagnosis and repair is either costly or in some cases, impossible.
2. *Few system parameters are observable*, all measurements come from sensors, which can be expensive, unreliable and invasive, monitoring is typically based on a small subset of the system parameters.
3. *The systems are dynamic*, the system exhibits time-varying behaviour, parameter values vary over a continuous range.

Automated process monitoring systems typically provide a set of alarms which are triggered whenever fixed thresholds are exceeded. Typical processes or systems can have over a thousand distinct alarms, and hundred of them can be activated in seconds. In such as situations, process operators may overlook relevant information, respond too slowly, panic when the rate of information flow is too great, and make incorrect decisions. This research is intended to contribute with an on line diagnosis algorithm for dynamic systems, as an aid to tackle these situations found in industrial processes and other applications.

1.2 Diagnosis problem

Monitoring and diagnosis of any dynamic system critically depend on the ability of estimate the system state given the observations. We are interested in dynamic hybrid system, systems which have both discrete and continuous

variables. A hybrid system consists of a set of discrete modes, which represent faulty conditions or operational modes of the system, i.e.*broken wheel*, *stuck valve*, *damaged pump*. Also a hybrid dynamic system has a set of continuous variables[2], i.e. *temperature*, *flow*, *speed*, *level*, etc. which model the continuous quantities that define the system performance.

We define the term *state* as the combination of a discrete mode plus a value of each continuous variable i.e {*broken wheel*, 85°*C*, 49 m/s}. Unfortunately, not all the hybrid system can be measured for several reasons, such as, high cost, impossibility or inability to observe the states. Therefore, we have a continuous observation function in order to compute the likelihood of an observation given the discrete mode and the values of the continuous variables. For both continuous variables and observations we are considering a stochastic component; also, for the discrete mode transitions. The learning model procedure for hybrid systems is a very important challenge; there is no practical solution yet. Additionally, we are looking for industrial applications in order to test our solutions.

Figure 1.1 shows a graphical representation of what we want to solve . Given only the observations (temperature) over time, we have to estimate/diagnose the most probable discrete mode (*normal operation, faulty fan, faulty grill, faulty fan and grill simultaneously, etc.*) and continuous state variable (no shown in this graph) on real time.

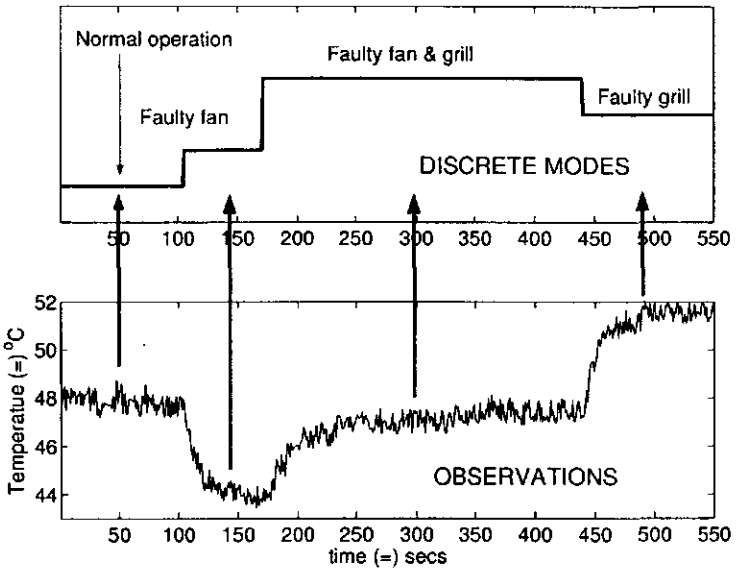

Figure 1.1: Problem definition. Given only the observations (lower plot) over the time, we want to estimate/diagnose the most probable discrete mode (upper graph) on real time.

1.3 Approaches to diagnosis dynamic systems

A variety of technologies have been proposed to date for the design of diagnosis systems based on the types of faults to be detected and the class of models deemed appropriate for these faults. Some popular approaches are:

- Analytical redundancy based fault detection and isolation systems
- Non-model based approaches such as those based on statistical hypothesis testing and signature analysis

[2]Commonly known as *state space* or *continuous state variable* in the engineering community.

- Fault tree methods, which have been studied in detail by reliability engineers
- Discrete events systems approaches
- Artificial Intelligence based model-based reasoning

our research is focused on the last approach. Artificial intelligence model-based diagnosis systems are generally based on a logical framework for diagnosis. These approaches generally are discrete, so they do not detect faults that appear as continuous variations. We can use monitors and quantize these continuous variables; however, they do not have enough resolution to give the practical results that industrial applications demand, specifically during the transient changes. Discrete techniques are not adequate for these limitations.

Typical fault detection and isolation systems can model continuous dynamic systems. These methods have the potential of detecting soft incipient faults even during the system's transient operations. However, they generally cannot work with hybrid systems. Some of the traditional *FDI* systems can be extended to hybrid systems; but, they are only useful under restricted conditions.

Monitoring and diagnosis of a hybrid dynamic systems depend mainly on the ability to estimate the hybrid state given the observable variables. Specifically for hybrid system, diagnosis task results computationally very expensive because the system has to follow multiple models and transitions between them. When the number of models grows, tracking all possible trajectories is exponential in the number of time steps and the problem then becomes intractable.

Qualitative reasoning techniques perform diagnostic inference involving multiple components in a computationally efficient manner, but they are limited by the low-resolution introduced by discretization of the continuous variables.

Some practical solutions to this intractable problem include approximation by Gaussians. Gaussian functions can summarize the distributions for each trajectory, but, the results are not always good. Some approaches with a similar idea (summarizing the information with few statistics) are based on bank of Kalman filters; however, Kalman filters as a Gaussian function are only applicable to continuous variables.

Our approach is based on a probabilistic technique known as *Particle Filtering* (*PF*). *PF* is a Sequential Monte Carlo method allow us to work with process densities with both continuous variables and discrete modes. The problem of estimating the state of a dynamic system from observations is named *Filtering*. As the state evolves, the system obtains a sequence of observations (and actions). Filtering estimates the state of the system as the posterior distribution, also known as *belief state*. *Bayesian filtering* reduces the filtering problem by assuming that the system state evolves in a Markovian way, so the past and future states are conditionally independent given the present state. The Markov property allows us to estimate the state recursively.

1.4 Particle Filtering

Several Particle filter algorithms were developed with different names. The most populars are:

- Monte Carlo filters, [Kitagawa, 1996]
- Sequential importance sampling, [Doucet, 1998]
- Bootstrap filters, [Gordon *et al.*, 1993]
- Condensation trackers, [Isard and Blake, 1996]

- Dynamic mixture models, [West, 1993]
- Survival of the fittest, [Kanazawa *et al.*, 1995]

A Particle Filter is a Markov chain Monte Carlo (*MCMC*) numerical algorithm that approximates the belief state using a set of samples, named *particles*, and maintains the distribution updated as new observations are made over time. Basically, the standard Particle Filter algorithm consists of two sequential steps:

1. *Sequential Importance Sampling Step.*
   A possible future state for each particle is generated by using the stochastic model of the system. Conditioning on the new information (observations) and the Bayes' rule, each particle is weighted by the likelihood of seeing the observations in the updated state represented by that particle.
2. *Selection Step*[3].
   To avoid the degeneracy of the sequential importance sampling, high-weight particles are replaced by several particles while low-weight particles tend to disappear.

Particle Filters (*PF*) have several features that make them a desirable algorithm for estimation/diagnosis. Some of these features are:

- Both discrete and continuous variables can be easily represented even with a single particle.
- *PF* are non-parametric and can represent a wide range of distributions. Specifically, non-linear systems with any type of prior belief distribution.
- The accuracy of results (quality of the approximation) can be traded for computational efficiency; basically, we can establish the number of particles based on how much computation time is available. The computational resources that *PF* demand depend on the number of particles. This feature allow us to implement them on real time applications.
- *PF* are easily implemented. The posterior distribution is specified through a set of particles.
- For many cases, the computational complexity is not affected as the dimension of problem increases.
- They can be easily implemented on parallel computers.

Although it has been shown by Monte Carlo simulations that these methods outperform the classical suboptimal methods, there are important drawbacks to resolve, specifically for diagnosis problems. Particle Filters have a well-known problem named *sample impoverishment.* In this problem, discrete modes with a non-zero probability of being the actual state of the system (because of the observations) contain no particles, and are then considered by the particle filter as zero-probability discrete mode. These are the discrete modes in which we are most interested, as faults usually show very low probabilities. This is one important problem that we have to cope with.

As the number of dimensions grow, the number of particles required for a good approximation grows exponentially. We can reduce the number of particles required by representing the continuous variables in a compact way (sufficient statistics, such as mean and covariance). The Rao-Blackwell equation [Casella and Robert, 1996] allow us to implement this idea. The Rao-Blackwell Particle Filtering (*RBPF*) algorithm is a highly efficient variant of Particle Filter which only samples the discrete modes and propagates the sufficient statistics for continuous variables.

[3]This step allows *PF* algorithms generate practical solutions

In Rao-Blackwell Particle Filtering, it is possible to sample the discrete modes directly from the posterior, and also it is possible to select the fittest particles before the sequential importance sampling step. These improvements result in an even more efficient algorithm called *look-ahead Rao-Blackwell Particle Filtering* (*la-RBPF*), [Morales-Menéndez *et al.*, 2002].

1.5 Related work

The most important contributions to the diagnosis/estimation of dynamic systems based on Particle Filtering have come mainly from researchers at:

- University of California, Berkeley
- Cambridge University
- Stanford University
- Carnegie Mellon University
- NASA Ames Research Center

In Chapter 2, I discuss the work of each one in detail, along with other important results. For the time being, I give a brief introduction to their related approaches.

Standard Particle Filtering has significant problems when sampling in high-dimensional spaces, and it can be inefficient. Sometimes, the model structure has special characteristics, and can be marginalized out analytically (*divide and conquer*). Rao-Blackwellization is a technique that allow us to marginalize some variables out and increase the efficiency of Particle Filtering. It reduces the size of the space over which we have to sample. The very first paper about Rao-Blackwellized Particle Filtering was presented at the Conference on Uncertainty in Artificial Intelligence in 2000. This paper presented important theoretical contributions and results based on work at *Cambridge* and *UC Berkeley*.

Stanford researchers have consolidated much research in the monitoring of dynamic systems using hybrid Dynamic Bayesian Networks (*DBN*s). Incorporating the unscented filter, a general framework for non-linear dynamic systems was developed and tested with one of the largest and most complex hybrid *DBN*s built to date. Important contributions have been presented on modelling and monitoring. On the monitoring side, they exploited some system properties, such as weak interaction, conditional weak interaction and sparse interaction, in order to implement inference algorithms with factored sampling. Factored sampling can help when working with huge domains.

In robotics, Particle Filtering implementations have had successful results[4]. Detecting and diagnosing faults is very important for these autonomous systems. *Carnegie Mellon* researchers have been working in this field, combining Particle Filtering algorithms with other ideas. Cost models have been incorporated into particle generation in order to account for the risk that arises relative to a control goal. A new Markov Decision Process algorithm that computes the risk function was developed. A state estimation method using Particle Filtering and principles of decision theory takes utility functions into account. Lately, probabilistic models, such as Partially Observable Markov Decision Processes (*POMDP*s) have been used for tracking the state (fault) of the stochastic system. Additionally, a hierarchical approach to fault identification has been incorporated.

Planetary rovers represent a challenge for the Artificial Intelligence community. *NASA*, in collaboration with many universities and research centers, has had many important results. Rovers and mobile robots present a special

[4]PF generated the first solutions for two previously unsolved problems (global localization and kidnapped robot)

problem for state estimation techniques due to a changing environment. Some proposed approaches combine continuous probabilistic state estimation using Kalman filters with discrete qualitative state estimation using *POMDPs*. Also, standard Particle Filtering had been used for hybrid diagnosis – this is the closest approach to ours. Lastly, the Gaussian Particle Filter (*GPF*), an efficient variant on the Particle Filtering family for non-linear hybrid systems, has been developed. In collaboration with *UBC*, an improved version, GPF_2, has also been developed. It includes features of *la-RBPF*, [de Freitas *et al.*, 2003], and has been tested in real domains with excellent results.

1.6 Contributions

Our contributions cover both a learning algorithm for the jump Markov linear Gaussian (*JMLG*) model parameters and look-ahead Rao-Blackwellized Particle Filtering (*la-RBPF*) for the diagnosis/estimation problem. Both were intensively validated with real data from different industrial applications.

- The main contribution of this research is the *la-RBPF* algorithm, which is an efficient variant of the Particle Filtering (*PF*) family. The most important *la-RBPF* features are low diagnosis error, low diagnosis variance and detection of discrete modes with very low prior probabilities. These features allow us to implement inference tasks on-line in real applications of hybrid dynamic systems. *la-RBPF* opens up many opportunities in different fields, including state estimation for control strategy selection, [Morales-Menéndez *et al.*, 2003], real-time diagnosis in mobile robots, [de Freitas *et al.*, 2003], etc. For domains with a large number of discrete modes, *la-RBPF* may be limited to off-line inference; however, there is still great potential due to the ever-increasing performance of computing hardware and software, or parallel computing.

- The *JMLG* model is a special case of a hybrid graphical model in which observable time series data are modelled in terms of unobservable discrete and continuous variables. This model gave good results for both modelling and inference in industrial processes. The proposed learning procedure for the *JMLG* model parameters, which combines the Least Squares Estimation and the Expectation-Maximization method, performed well and allowed us to capture the dynamic behaviour of processes operating in different conditions or under faulty situations. Certainly, the *JMLG* model is an efficient and practical way to model hybrid dynamic systems.

1.7 Outline

Chapter 2 provides an overview of the main approaches to diagnosis in dynamic systems. Briefly, we define the kind of domains we are considering. Some definitions commonly accepted by the control engineering community are presented. The most important approaches to tackling the fault diagnosis problem in dynamic systems are briefly discussed. We review traditional model-based methods where State Observers, Parameter Estimation, Kalman Filters, and Parity Space are the most representative. Then, typical Artificial Intelligence techniques that have been applied to diagnosis in dynamic systems are discussed. We consider Neural Networks, fuzzy logic, qualitative reasoning, logic-based, and Dynamic Bayesian networks. Finally, related work in Particle Filtering (*PF*) is presented. Based on these approaches, we motivate our research.

In Chapter 3 we provide the fundamental tools on which our research is based. We discuss Hidden Markov Models (*HMMs*) and State-Space Models (*SSMs*). *HMMs* and *SSMs* are well-known models; however, most real and interesting processes cannot be characterized by either purely discrete or purely linear-Gaussian dynamics. The

Jump Markov Linear Gaussian *JMLG* model evolves as a natural generalization of *HMMs* and *SSMs*. Learning the *JMLG* parameters is still an intractable problem; we show the Variational Learning approach, the Expectation-Maximization (*EM*) method and Least Squares Estimation (*LSE*) as the building blocks of our proposal to cope with this problem. Finally, the standard Particle Filtering and Rao-Blackwellized Particle Filtering algorithms are presented. Detailed descriptions of each procedural step are shown. Our proposal is based on these methods.

Our proposal is presented in Chapter 4. We divided this chapter into the modelling problem and the inference (estimation/diagnosis) problem. First, we describe the Jump Markov Linear Gaussian model for hybrid dynamic systems. Based on this model we propose a learning algorithm. The learning algorithm is an iterative procedure based on the *LSE* and *EM* methods. Next, we show the proposed algorithm for the inference problem, *look-ahead Rao-Blackwellized Particle Filtering* (*la-RBPF*). We end this chapter with some comments on the main results for the learning and *la-RBPF* algorithm.

The experimental work is one of the most important contributions of this research. Intensive experiments were conducted at ITESM Campus Monterrey (for the industrial processes), and at the University of British Columbia (for the mobile robot). Chapters 5 and 6 show these results. We tested both the learning and inference algorithms in four experimental domains and two simulated systems. Chapter 5 presents the modelling and estimation/diagnosis results for the industrial dryer, level tank, heat exchanger, and mobile robot. Each domain contains interesting and distinctive dynamic characteristics. In order to test our inference algorithm with more complex systems, in Chapter 6 we present a non-linear Continuous Stirred Tank Reactor (*CSTR*) simulator. The *CSTR* simulator allowed us to conduct tests involving more discrete modes, more continuous variables, more observations, etc. Additionally, using the effective *JMLG* models we learned for the mobile robot, we generated some simulated runs involving more discrete modes and different noise levels. These results are also presented in Chapter 6.

Finally, in Chapter 7 we present a detailed discussion of the learning and inference algorithms based on the experimental results. We present our main conclusions, organized into contributions and limitations for both the learning and *la-RBPF* algorithms. Some significant real applications of *la-RBPF* at NASA Ames Research Center, [de Freitas *et al.*, 2003], are discussed. We end this chapter with some open questions and future directions.

Chapter 2

Diagnosis in Dynamic Systems

2.1 Introduction

For years, the only way to learn about faults in industrial processes was through our senses; looking for changes in color, listening for unusual sounds, feeling for changes in temperature or smelling for fumes. Later, measuring instruments appeared which could provide numerical information about the processes. However, these instruments can also show faults. The impact of these faults becomes more critical if the instruments are used in automatic control systems; malfunctions without the presence of human operators can be catastrophic; see chapter 1, section 1.1.

The operation of industrial processes requires increasingly advanced *supervision* and *fault diagnosis* (and fault tolerance) to improve cost efficiency, availability, reliability and safety. Supervisory functions enable operators to identify unacceptable process states, to make better decisions and to take appropriate actions [Enste and Uecker, 2000]. Fault tolerance and fault diagnosis are becoming more and more important.

Fault tolerance can be achieved by either *passive* or *active* strategies. The passive approach makes use of robust control techniques to ensure that the system becomes less sensitive with respect to faults. In contrast, the active approach provides fault accommodation, i.e. the reconfiguration of the system when the fault occurs.

As in many areas, computers have made a huge contribution and have taken automatic systems to a new level. Currently, computers are operating complex systems in almost every domain with excellent results. Features such as massive-information integration and high speed processing make it feasible to efficiently detect and diagnose many processes. Initially, industrial computers were only applied to complex processes expensive enough to justify their use, but they have now become part of our computerized society.

We start this chapter with some basic definitions of dynamic systems and fault diagnosis. Based on these definitions and a simple example, we roughly describe the kind of dynamic process which we want to work with. A general review of the main approaches to fault diagnosis in dynamic systems is presented. Then we discuss the main applications of Particle Filtering in fault diagnosis. We end this chapter with a brief justification of our research.

2.2 Definitions

In loose terms a *system* is an object in which variables of different kinds interact and produce observable signals. The observable signals that are of interest to us are usually called *outputs*. The system is also affected by external stimuli. External signals that can be manipulated by the observer are called *inputs*. Figure 2.1 shows a system with

one input $u(t)$ and one output $y(t)$, called a SISO (single-input single-output) system. A MIMO (multiple-input multiple-output) system has n_u inputs $u(t)$ and n_y outputs, Figure 2.2.

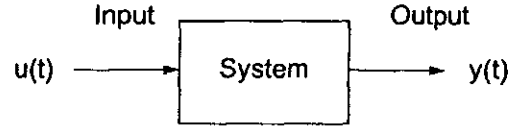

Figure 2.1: SISO dynamic system. One input $u(t)$ affects a single output $y(t)$.

The dynamic systems that we are considering have three important properties:

- *Linearity.* They are linear systems. For non-linear dynamic systems a linear approximation is taken.
- *Time invariance.* The relationship between the inputs and outputs is permanent; it does not vary with time. In the mathematical representation of the system, this is equivalent to the model parameters being constant.
- *Causality.* The system output at any time t depends only on inputs at time t and before; it is not affected by future inputs.

Causality is a natural property of all physical systems. Linearity and time invariance do not always hold; however, in some cases they may be assumed in order to simplify the mathematical treatment. Other important definitions to be considered are given in Appendix A, section A.1.

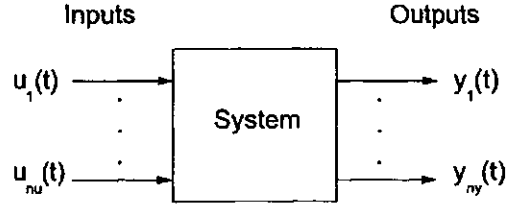

Figure 2.2: MIMO dynamic system. n_u inputs $u(t)$ affect n_y outputs $y(t)$.

Terminology in the fault diagnosis field is not consistent; however, there are some commonly accepted *states and signals* definitions (taken from the IFAC Technical Committee SAFEPROCESS, [Isermann, 1997b]):

- *Fault, Failure and Malfunction.*
  A *fault* is a non-permitted deviation of at least one characteristic property or parameter of the system from the acceptable/usual/standard condition. Faults can differ in terms of time dependency, as shown in Figure 2.3. A *failure* is a permanent interruption of a system's ability to perform a required function under specified operating conditions. A *malfunction* is an intermittent irregularity in the fulfilment of a system's desired function.

- *Disturbance and Perturbation.*
  A *disturbance* is an unknown (and uncontrolled) input acting on a system. A *perturbation* is an input acting on a system, which results in a temporary departure from the current state.

- *Error and Residual.*
  An *error* is a deviation between a measured or computed value of an output variable and the true, specified or theoretically correct value. A *residual* is a fault indicator, based on a deviation between measurements and model-equation-based computations.

- *Symptom.*
  A *symptom* is a change in an observable quantity from normal behaviour.

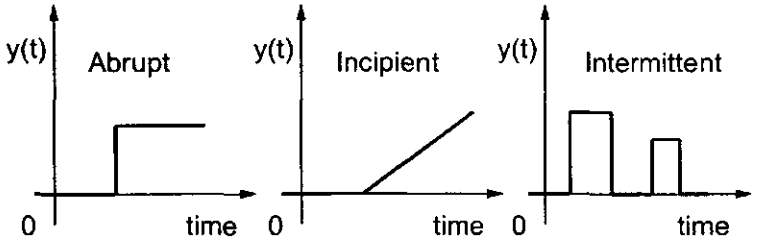

Figure 2.3: Time-dependency of faults. $y(t)$ represents the output signal, and t is the continuous time. y(0) represents the output variable in acceptable condition. The plots show different ways in which faults can appear.

We can classify [Gertler, 1998] the faults we are interested in into three categories:

- *Process faults.* These can be divided into *additives* and *multiplicatives.* Additive faults are unknown inputs acting on the process which are normally zero and which, when present, cause a change in the process outputs of known inputs; see Figure 2.4. Multiplicative faults are changes (abrupt or gradual) in some process parameters. They cause changes in the process outputs which depend also on the magnitude of known inputs.

- *Sensor faults.* These are discrepancies between the measured and actual values of individual process variables. These faults are usually considered additive (independent of the measured magnitude), although some sensor faults, such as sticking or complete failure, may be better characterized as multiplicative.

- *Actuator faults.* These are discrepancies between the input command of an actuator and its actual output. Actuator faults are usually handled as additive, although again, some kinds (sticking or complete failure) may be better described as multiplicative.

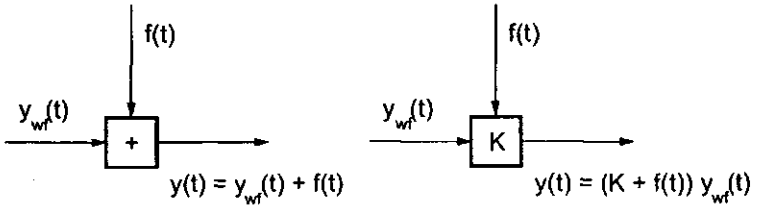

Figure 2.4: Basic models of faults: additive and multiplicative. $y_{wf}(t)$ is the fault-free output signal, $f(t)$ is the fault, and $y(t)$ is the observable output signal.

Fault detection and diagnosis systems are commonly defined as:

- *Fault detection*, the determination that something is going wrong;
- *Fault isolation*, the determination of the exact location of the fault;
- *Fault identification*, the determination of the dimensional, spatial, and temporal location of the fault, and the category of fault.

Fault diagnosis is the term usually used for both the isolation and identification tasks. *Fault detection* and *fault isolation* are essential in any practical system, but in some domains *fault identification* may not be worth the extra effort it requires. Usually, we consider [Leonhardt and Ayoubi, 1997] automatic fault diagnosis as two sequential steps: *fault detection* and *fault diagnosis*. However, most practical systems contain only the *Fault Detection and Isolation* tasks and are labelled *FDI* systems [Gertler, 1998].

Certain assumptions are normally adopted for the fault detection and diagnosis task. Any noise generated from the process, sensors or actuators is considered random with zero mean; any nonzero mean is a fault or disturbance. It is assumed that *faults* are not present initially in the system, but that they arrive later.

We can measure the *detection performance* by three different means:

1. *Fault sensitivity*, the ability to detect small faults.
2. *Reaction speed*, the ability to detect faults almost on time (small delay).
3. *Robustness*, the ability to operate in the presence of noise, disturbances and modelling errors.

2.3 Scope

The main goal of this research is to do real-time diagnosis and state estimation (as an inference task) in dynamic industrial processes. By *diagnosis* we mean the detection/determination of faults that could occur in the process itself, in its measuring instruments or in its actuators. By *state estimation* we mean the identification of different operating conditions which the process can be in. For *dynamic industrial processes* we consider boilers, reactors, power plants, oil refineries, chemical plants, etc., and well as machines like robots, rovers, vehicles, etc. A robot or a chemical plant can have hundreds, even thousands of variables; however, we can divide the system into small practical subsystems and tackle the problem in an isolated fashion. We are exclusively concerned with *dynamic systems* characterized by *continuous-time* operation. If the dynamic system has non-linear behaviour, it has to be modelled as a series of linear segments. In the following subsection, we show an example of the kind of process that we are interested in.

2.3.1 A continuous dynamic system

Let us consider a multiple-input multiple-output continuous dynamic system, as defined in Appendix A, section A.1. As an example, we consider a heat exchanger[1]. In this heat exchanger a process stream is heated by condensing steam. The goal of this industrial equipment is to heat the process fluid from some inlet temperature up to a specific outlet temperature, Figure 2.5.

The outlet temperature is defined as the output variable, $y(t)$, in this system. The latent heat of condensation of the steam provides the required energy to the process fluid. The steam flow corresponds to the input variable, $u(t)$. Considering only these definitions, the natural multiple-input multiple-output heat exchanger can be visualized as a single-input single-output continuous dynamic system, Figure 2.6.

[1] One of the experimental processes used to test our algorithms was a heat exchanger.

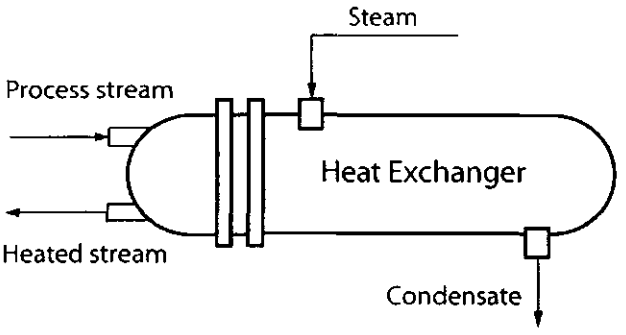

Figure 2.5: Heat exchanger. By condensing steam, the process stream is heated from some inlet temperature up to an outlet temperature.

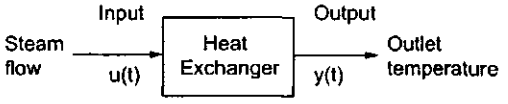

Figure 2.6: Single-input single-output continuous dynamic system. The outlet temperature is the output variable, and the steam flow is the input variable

In order to control the outlet temperature, we have to measure it. We can use a temperature sensor/transmitter (usually called primary/secondary control elements). If there exists a difference between the measured temperature and the desired one, some action must be taken to correct this deviation. The steam flow has a direct effect on the outlet temperature; indeed, it is the most influential variable in changing the outlet temperature. The steam flow can be manipulated by a control valve (called a final control element). There are additional devices required for proper operation of this system: steam traps, pumps, vacuum breakers, and sensors/transmitters (flow, pressure, temperature, concentration, etc). Some of these additional devices are shown in Figure 2.7 and described in Table 2.1. However, many devices are omitted for clarity.

Table 2.1: Heat exchanger instrumentation.

tag − name	Functional name	Description
FT10	Flow sensor/transmitter	Process stream flow
TT10	Temperature sensor/transmitter	Process stream temperature
TT20	Temperature sensor/transmitter	Heated stream temperature
FT30	Flow sensor/transmitter	Steam flow
PT30	Pressure sensor/transmitter	Steam pressure
FV30	Flow valve	Steam flow valve

Finally, considering this heat exchanger as part of a larger process, some basic relationships with other subsystems (steam boiler, steam users, previous subprocess, following subprocess, etc.) are shown in Figure 2.8. As we can see, there are many variables that can affect the outlet temperature.

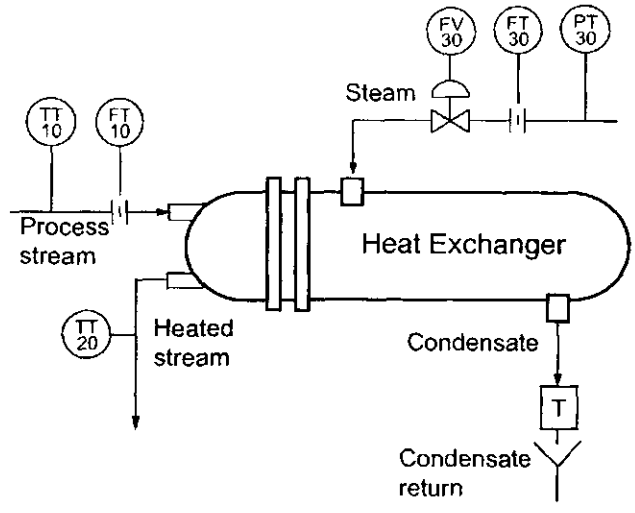

Figure 2.7: Instrumented heat exchanger. Basic instrumentation is required for monitoring and control purposes.

Based on the described process, we can identify three very simple possible faulty points (although there exist many more):

- Pumps
- Sensors/actuators
- Steam trap

Pumps and sensors/actuators have well-known roles in various systems, and the effect of faulty conditions is easily inferred. We provide a brief description of a steam trap.

A steam trap removes condensate, air and CO_2 from the heat exchanger, as fast as they form. The steam trap enhances the system's overall efficiency and reliability, and results in substantial energy savings. A faulty steam trap can generate problems such as loss of live steam, dirt and scale formation, and steam flow blocking.

It is important to note that we are not trying to find the cause of a faulty device itself. For example, for a faulty pump, such causes could include: starter defective, shaft broken, intake blocked, discharge valve closed, speed too low, etc. There are specific procedures for these problems which are generally provided by suppliers. We are interested in the devices as parts of the system, and in how their faulty conditions can be manifested in the variables that we can observe.

In addition, different (non-faulty) operating conditions can be considered for this system. Three examples are shown:

- *Normal operating conditions.* Every subsystem works properly. The variables are at their nominal operating points.
- *High process flow.* Some subsystems (Subsystem-1, Subsystem-2, etc.) are demanding more heated process flow than normal.
- *Low steam flow.* Some steam users are demanding more steam than normal, so we have less available steam.

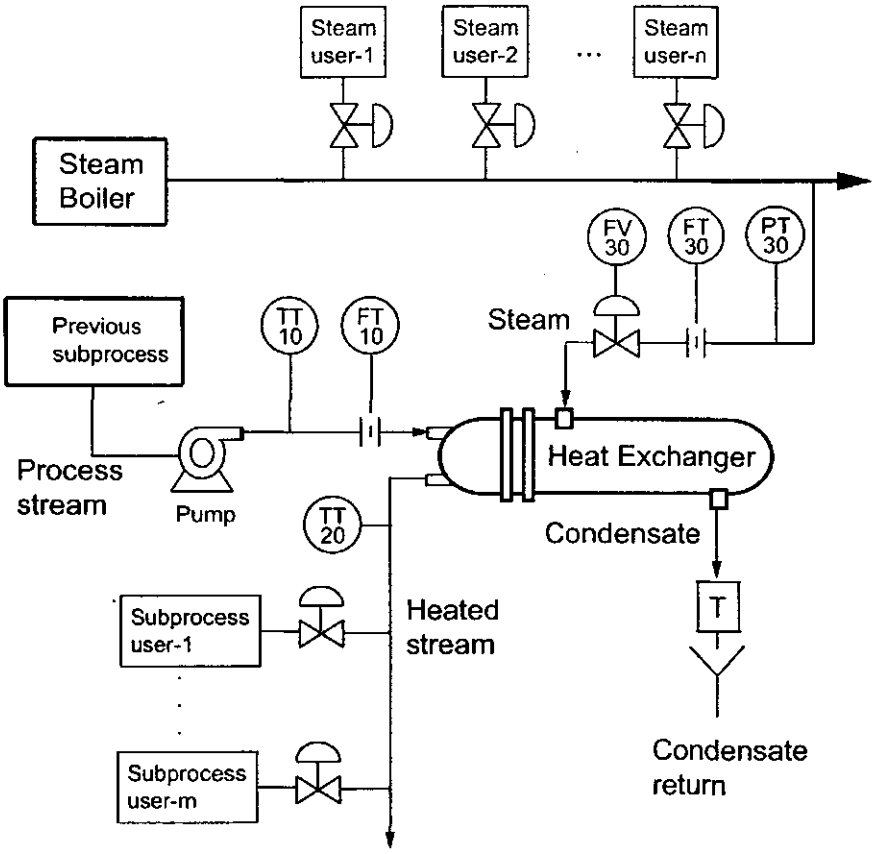

Figure 2.8: Heat exchanger and other subsystems. The outlet temperature can be affected by many variables.

Each situation has different dynamic behaviour, consequently their variables show different values. In our original control problem, a new policy is required for each one. If the operating condition can be successfully estimated, the control policy is straightforward. State estimation is a practical inference task in many domains.

Every faulty or non-faulty operating condition has to be characterized as a continuous dynamic system, for which the only information we have consists of noisy observations (input/output variables). We must perform our inference task (diagnosis or state estimation) based on these observations.

It is important to note that discrete devices are part of this system. By discrete devices we mean devices which exhibit only discrete values such as pumps {on,off}, solenoid valves {open, closed}, level sensors {low, medium, high}, and alarms {low-low, low, high, high-high}. However their impact only is considered through the continuous observations. We work with any process that can be modelled as a continuous dynamic system.

2.4 Fault diagnosis in dynamic systems

Methods for *fault detection and diagnosis* may be roughly classified into *model-free methods* and *model-based methods*, [Gertler, 1998]. We present a brief description of both approaches.

2.4.1 Model-free methods

The following *FDI* methods do not use the mathematical model of the process:

1. *Physical redundancy.* Multiple sensors are installed to measure the same variable; the discrepancies indicate a sensor fault.
2. *Special sensors.* Some sensors are installed explicitly for detection and diagnosis, e.g. a maximum temperature sensor in some device, process or system.
3. *Limit checking.* Process measurements are compared to preset limits. Exceeding the threshold indicates a fault situation.
4. *Spectrum analysis.* Most process variables exhibit a frequency spectrum under normal operating conditions; any deviation from the normal operating condition is a symptom of abnormality.
5. *Logic reasoning.* These techniques evaluate the information obtained from the detection of conditions. The simplest techniques consist of trees of logical rules of the *"IF-symptom-AND-symptom-THEN-conclusion"* type. These techniques are complementary to the methods outlined above.

Statistical Process Control (SPC) charts and statistical *Hypothesis Testing (HT)* have also been used in fault detection and isolation systems. See [Himmelblau, 1978] for SPC in chemical processes and [Duyar and Eldem, 1992; Rudas, 1991] for HT in various applications.

[Morales *et al.*, 2001] proposed an *FDI* system where parametric estimation is used for residual generation. The resulting residuals are processed by a statistical decision block, where hypothesis testing and statistical process control techniques are used for determining the presence and the location of a fault.

2.4.2 Model-based methods

Model-based *FDI* uses a mathematical model of the process. Continuous-time dynamic systems are modelled by differential equations or equivalent transformed representations. However, computers operate in a sampled domain, using sampled data. For practical reasons, we describe the process in discrete time, in the form of difference equations.

Most model-based *FDI* methods follow the *analytical redundancy* concept. Real measurements of a process variable are compared to analytically calculated values of the same variable. The resulting differences, named *residuals*, are indicative of faults in the process. This generation of residuals needs to be followed by a decision-making block in order to isolate and identify the faults, Figure 2.9, [Gertler, 1998].

There are four main classical methods for residual generation in model-based *FDI* approaches:

1. *Diagnostic observers.* Observer prediction errors are used as fault detection residuals.
2. *Kalman Filter.* The prediction error of the Kalman Filter [Kalman, 1960] can be used as a fault detection residual.
3. *Parity (consistency) relations.* Parity relations are rearranged direct input-output model equations, under linear dynamic transformation. The transformed residuals are used for detection and isolation [Frank, 1990].
4. *Parameter estimation.* A reference model is obtained by first identifying the process in a fault-free situation. Then the parameters are re-identified on-line. Deviations from the reference model are used as a basis for detection and isolation.

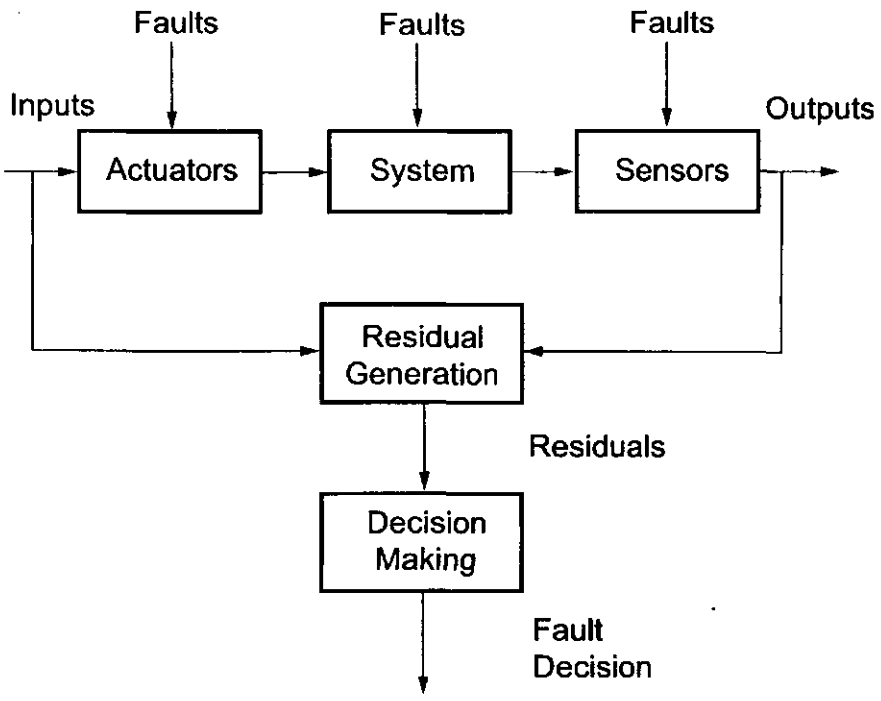

Figure 2.9: Fault detection and isolation system. Residuals are generated by comparison between real values and analytically computed values. This must be followed by a decision-making block in order to isolate and identify the faults.

2.5 Approaches

Model-based methods are somewhat overlapping approaches; it is quite difficult to define a precise classification with specific characteristics. All of them combine similar ideas and sometimes similar principles.

Classical approaches in the control engineering community can be simply grouped into *State observers*, *Parameter estimation*, *Kalman filter* and *Parity space*. These have made huge contributions in different domains with successful results.

There is another important group of approaches which are based on typical Artificial Intelligence techniques. They can be classified into *data-based* models and *knowledge-based* models.

Data-based models constitute an alternative to the analytical approach for *FDI* systems. In many practical applications, there are large archives of process data which can be used to set up data-based models. The two most common types of data-based models are *artificial neural networks* and *fuzzy* relational models. There are also *neural-fuzzy* approaches which combine their main features.

In the case of fault diagnosis in complex systems, we are faced with the problem that no, or no sufficiently accurate, mathematical models are available. The use of knowledge-based techniques in the framework of diagnosis expert systems or in combination with a human expert is then the only feasible way. The knowledge-based approach uses the available knowledge to derive either a qualitative description of the system in the form of a *qualitative* model, or a *rule-based* representation.

Rule-based techniques make use of heuristic symptoms, knowledge about the process history, or statistical

knowledge, the evaluation of which is organized in the framework of diagnostic expert systems. If the symptoms are considered in connection with the inputs of the system, one can speak of a symptom-model-based approach. The major difficulty is the knowledge acquisition, which is known as an extremely difficult task.

Powerful diagnostic frameworks have been developed based on logic, probability and the combination of both. We present examples of two approaches, *logic-based* and *dynamic Bayesian networks.*

Finally, in the following section we present the most representative applications of Particle Filtering to fault diagnosis, on which our research is based. Most of the approaches that we describe have been successfully applied in various fields; we mainly address their application to fault diagnosis in continuous dynamic systems.

2.5.1 State Observers

The *state* is the minimum set of variables which describes the behaviour system [Ogata, 1995]. Observer-based fault detection techniques have been applied to complex processes including nonlinear and time-varying systems with considerable modelling uncertainty. The basic idea is that one reconstructs measurements of the process using a state-observer, and makes the decision on possible faults in the process. In contrast, the parameter estimation approach (section 2.5.2) makes the fault decision by performing an on-line parameter estimation. Both methods are complementary and are therefore best applied in combination. The parity space approach (section 2.5.4) leads to certain types of observer structures and is therefore structurally equivalent, even though the design procedures differ [Frank and Ding, 1997].

Basically, in the observer-based approach, the generation of residuals reflecting the faults is done by estimating outputs of the process and using the estimation errors as the residuals. For fault detection, a simple observer is sufficient, whereas for the localization of faults, properly structured sets of residuals are required. In doing this one may use linear or nonlinear, full or reduced-order, fixed or adaptive observers [Frank and Ding, 1997]. There are many different approaches for the design of diagnostic observers. Examples of these are geometric methods, spectral theory, frequency domain and algebraic methods [Frank *et al.*, 2000].

[Chen *et al.*, 2001] designed a novel approach to robust fault diagnosis, modelling uncertainties and unknown input disturbances for discrete systems. (The approach is also applicable to dynamic systems.) First, they decouple the unknown input disturbances in the system. Then, a state observer is constructed and the faults are identified by the least squares method, based on the relationship between the faults and state observation error. Finally, the robustness of the identification algorithm to modelling uncertainties is enhanced using the dead zone function.

Robustness is essential for practical applications. Many theoretical articles about the robustness problem related to modelling uncertainty have appeared [Frank, 1992]. There is a huge theoretical foundation for the analytical observer-based approach in time domain and linear systems. Also, there are extensions to the frequency domain design of linear observers, adaptive observers, and nonlinear robust observers [Ding *et al.*, 1994; Ding *et al.*, 1990; Ding and Frank, 1999; Caccavale, 1998].

2.5.2 Parameter Estimation

Parameter estimation was developed for and widely used in modelling, signal processing and control. It can also be applied to fault detection and isolation systems [Isermann, 1997a]. The main idea of parameter estimation in *FDI* systems is the on-line estimation of the parameters of the actual process, followed by comparison to their nominal values. The resulting deviations form the residuals used for *FDI* [Isermann, 1997a]. To interpret the faults, we need to explain the deviations in terms of physical parameters. Hence the model should be as detailed as possible. This effort produces the following advantages:

- it provides a deeper insight into the process,
- if the relationship between the model parameters and physical parameters is unique, fault isolation is easy to implement, and
- it provides direct fault identification.

The main limitation is that the estimated parameters must be persistently excited by an input signal. Also, to isolate faults, the relationship between the physical and mathematical parameters must be unique. This represents another important potential limitation for complex industrial processes.

[Dinca *et al.*, 1999] describes a model-independent probabilistic approach which can be used for the estimation of system parameters and unmonitored state variables. These can include noisy data and/or parameter/modelling uncertainties. This approach can estimate the probability of finding the system in specified intervals of system parameters and dynamic variables. This is good for diagnostic purposes because it allows us to rank the possible system faults in terms of their likelihood through a set of logical rules. These rules map the relationships between system parameters and variables to system faults. The estimation methodology is based on a Markovian representation of system dynamics.

2.5.3 Kalman Filters approach

Kalman filtering (*KF*) [Kalman and Bucy, 1961] is a well-known technique for state and parameter estimation. For processes with linear dynamics and Gaussian noise, it provides an excellent means of tracking system states. Kalman filtering is a recursive estimation procedure that uses sequential measurement data sets. Prior knowledge of the state, expressed by the covariance matrix, is improved at each step as prior state estimates are used for prediction and new measurements for subsequent state update.

A bank of Kalman filters was first used by [Magill, 1985], as a parallel set of estimators for a sample stochastic process. [Hanlon and Maybeck, 2000] improved the technique, calling it Multiple Model Adaptive Estimation (*MMAE*) and using it to reliably detect and identify sensor and actuator failures in aircraft. *MMAE* consists of a bank of parallel *KF*s, each with a different model, and a hypothesis testing algorithm. Each of the Kalman filters can be identified by a discrete value of a parameter vector. Each filter is provided a measurement vector and the input vector, and produces a state estimation and a residual. The hypothesis testing algorithm uses the residuals to compute conditional probabilities of the various hypotheses that are modelled in the *KF*s, conditioned on the history of measurements received up to that time, and to compute an estimate of the true parameter vector. A further idea considers a *KF* bank, named the residual correlation *KF* bank (*RCKFB*), with outputs that are estimates of the power spectral density of each of residuals. Using the spectral content of the residual, the fault can be detected.

[Goel *et al.*, April 2000; Roumeliotis *et al.*, May 1998; Rudas, 1991] used multiple parallel *KF* estimators to model the system behaviour under each type of fault in a mobile robot. A similar application for mobile robots, but based on the Interacting Multiple-Model (*IMM*) approach was proposed by [Hashimoto *et al.*, Oct 29 Nov 03 2001]. The *IMM* algorithm requires specification of an entire matrix of failure state transition probabilities for its Markov model, which often is significantly more difficult for a designer to provide accurately than a single lower bound. For this and other reasons, [Fisher and Maybeck, 2002] proposed *MMAE* with filter spawning [Fisher, 1999] to detect and estimate faults on actuators (VISTA F-16). Filter spawning is used to include additional filters with different hypotheses in the MMAE bank.

[Liu and Si, 1997] proposed a full-order observer to detect and isolate multiple faults. A fault isolation filter was designed such that faults can be asymptotically detected and isolated. The observer's gain matrix is determined

so that the ith component of the output residual is decoupled from all but the ith fault. To satisfy this property, the columns of the fault detectability matrix are assigned as eigenvectors of the observer's transition matrix with a set of fixed eigenvalues. [Keller, 1999] proposed extending this idea to detect faults appearing simultaneously or sequentially in discrete-time stochastic linear systems. The obtained fault isolation filter is very similar to the predictor-corrector structure of the standard Kalman filter, allowing the establishment of its convergence and stability conditions.

[Washington, 2000] presented MAKSI (Markov and Kalman State Identification) which combines continuous probabilistic state estimation using Kalman filters with discrete qualitative state estimation using a Markov-model representation, applied to rovers. Kalman filters can track multiple hypothesis; however, they lack methods for automatically choosing which states to track. Tracking all possible states is infeasible in practical applications. *KF*-based localization has also become common practise in robotics [Goel *et al.*, 1999]. Kalman filters have been used in combination with many approaches; the Kalman filter is fully discussed in Chapter 3.

2.5.4 Parity space approach

This method generates residuals using analytical information [Patton and Chen, 1991]. Residuals are formed as the differences between actual process outputs and those predicted by the model. These residuals are then subjected to a linear transformation to obtain the desired fault-detection and isolation properties. To enhance fault isolation, residuals are designed to exhibit directional or structural properties in response to particular faults. The residuals must present some dynamic characteristics, for the desired transient behaviour and noise filtering. With parity relation design, these specifications are explicit, though they may need to be relaxed to accommodate the natural requirements of causality and stability of the residual generator [Gertler, 1997]. Among the model-based *FDI* techniques, the parity space method is particularly appealing due to the lack of stability concerns common to other methods [Medvedev, 1995].

[Staroswiecki and Comtet-Varga, 2001] extends the parity space approach to non-linear systems, focusing on the issues of robustness, fault detectability and isolability. The approach was illustrated using a (simulated) induction motor. However, for practical applications, calculation of the residuals can be intractable.

2.5.5 Artificial Neural Networks approach

Artificial Neural Networks (*ANN*) are non-linear systems. The non-linear transformation between inputs and outputs results from their inner structure. Artificial neural networks consist of neurons which are activated as soon as their inputs exceed a certain threshold. The neurons are arranged in layers which are connected such that the signals at the input are propagated through the network to the output. The choice of the transfer function of each neuron yields the overall non-linear behaviour of the network. During a training period, a set of neural network parameters is learned from a given set of data, aiming at the best approximation of the behaviour of the system [Koppen-Seliger and Frank, 1995].

Fault detection and diagnosis can be viewed as a pattern association problem where we wish to recognize deviations, disturbances or malfunctions and associate them with their respective causes. Artificial neural networks can store knowledge by learning from historical fault information. During learning the network memorizes the fault patterns, and in the testing stage, when shown a new input pattern, it associates that pattern with the appropriate memorized one. Some artificial neural networks can also interpolate the memorized patterns.

There exist a variety of architectures such as the *feed-forward* and *recurrent ANN*. *Feed-forward* networks are commonly utilized for pattern recognition tasks, while *recurrent* networks characterize nonlinear dynamic feed-

back systems. For fault detection, *feed-forward* networks trained using the error back propagation algorithm [Rumelhart *et al.*, 1986] are the most popular.

[Fan *et al.*, 1993] showed that networks trained with the error back propagation algorithm can often successfully diagnose new cases of faults. However, there are some difficulties such as entrapment into a local minimum during network training, long training times, and the need for several runs to optimize structural and functional parameters of the network.

Radial basis function and *adaptive resonance theory* networks have also been employed for fault detection and diagnosis, [Whiteley and Davis, 1994]. Neural networks based on *adaptive resonance theory* possess the needed *stability-plasticity*[2]. *Radial basis function* networks are good for extrapolation; however, they do not always guarantee an optimal solution to classification problems.

[Vora *et al.*, 1997] proposed a *counter-propagation* neural network for performing fault diagnosis. This kind of network has several advantages; mainly, the training algorithm is simple (no entrapment into local minima), and the optimal network architecture can be determined beforehand. However, *counter-propagation* neural networks have severe limitations for multiple faults. [Watanabe *et al.*, 1994] proposed a *Hierarchical ANN* for multiple fault diagnosis with notable success.

For residual generation purposes the neural network simply replaces the analytical model describing the process under normal operation. Employing a non-linear input/output description, the neural network approximates the non-linear process. This general pattern may be substantially compressed by estimating each system's output with a separate neural network. Not all of the system outputs may be influenced by the faults under consideration. The appropriate choice of input space is one of the most difficult tasks in configuring the neural network. One needs to have sufficient knowledge to use a trial and error strategy, or one must apply an optimization algorithm [Koppen-Seliger and Frank, 1995].

The major difficulties in the application of neural networks to *FDI* schemes is the lack of analytical information on the performance, stability and robustness of the neural network, [Polycarpou and Vemuri, 1995]. Some applications are [Terra and Tinos, 2001] on robotic manipulators, and [Goel *et al.*, April 2000] on mobile robots.

[Valles, 2001] proposed a supervisory scheme in adaptive control systems based on *FDI* techniques. These supervisory schemes combined neural networks, state observers, statistical hypothesis testing and statistical process control charts.

2.5.6 Fuzzy Logic approach

Residual evaluation can be seen as a classification problem [Frank and Koppen-Seliger, 1997]. The task is to match each pattern of the symptom vector with one of the preassigned classes of faults or the fault-free case. Fuzzy logic [Zadeh, 1978], in this context, means how well a variable, a fault, or an operational condition satisfies a vague description. The principle of residual evaluation using fuzzy logic consists of a three-step process:

1. The residuals have to be *fuzzified*.
2. The residuals have to be *evaluated* by an inference mechanism using fuzzy IF-THEN rules.
3. The residuals have to be *defuzzified*.

The *fuzzification* of the residuals is a mapping of the representation using crisp values into a representation by fuzzy sets. It is a fuzzification of the threshold [Kiupel and Frank, 1993]. Residuals must meet the ideal conditions of being zero in the fault-free case and different from zero in the case of a fault.

[2]Stability: does not forget the learned information. Plasticity: ability to learn new patterns

The task of *fault decision* is to infer a fault (in the set of possible faults) from a set of residuals. In this case, the residuals are defined by their fuzzy sets, and the relationships between the residuals and the faults are given by IF-THEN rules. We can solve this task with the aid of a fuzzy relation, defined by the theory of fuzzy logic. We can transform the set of fuzzified residuals onto the set of fuzzified statements of faults.

Finally, the fuzzy information on the faults has to be converted into crisp sets. Many *defuzzification* algorithms are known [Montmain and Gentil, 1993]. Fuzzy logic can be applied to threshold adaptation with great success in the case of uncertain systems with changing operating conditions [Schneider, 1993].

As an example of this approach, consider the fuzzy-reasoning application [Dash *et al.*, 2003] in fault diagnosis. The fault diagnosis task is based on patterns presented in the measurements process. The temporal patterns that a process event generates can be used to infer the state of operation using a pattern-matching approach. However, noise and qualitative properties of the features could lead to imprecise classification boundaries at the trend-identification stage, and hence at the trend-matching stage too. Moreover, exact inference could require a huge knowledge-base of patterns with no guarantee during the classification stage. Fuzzy reasoning can ensure robustness to the uncertainty in the trends' identification and matching stages. The main motivation for using fuzzy inferencing to perform fault diagnosis is to handle the imprecisiness in trend representation. Basically, fuzzy logic is used to exploit the trend information generated by an interval-halving strategy [Dash *et al.*, 2001] for trend extraction. This application was tested in the fault diagnosis of a reactor simulator[3].

Fuzzy logic has been implemented in very successful commercial applications, particularly in control systems, so *FDI* based on Fuzzy logic is naturally ideal in many domains.

2.5.7 Qualitative Reasoning approach

The artificial intelligence community developed the theory of diagnosis from first principles (system structure and behaviour descriptions). The early important works on static systems were [Davis, 1984; Genesereth, 1984; de Kleer and Williams, 1987; Reiter, 1987]. Later, [Forbus, 1984; de Kleer and Brown, 1984; Kuipers, 1984] [Kuipers, 1986] developed the theory of Qualitative Reasoning (*QR*) about physical systems, which was motivated initially by diagnostic problems. Using Qualitative Simulation [Kuipers, 2001], it was possible to supervise simple dynamic processes and diagnose their faults. Pioneer applications are [Forbus, 1986; Kuipers, 1987; Dvorak and Kuipers, 1989; Ng, 1990].

Qualitative Simulation predicts the set of possible behaviours consistent with a qualitative differential equation model of the world. Its value comes from the ability to express natural types of incomplete knowledge of the world, and the ability to derive a provably complete set of possible behaviours in spite of the incompleteness of the model.

A Qualitative Differential Equation (*QDE*) model [Kuipers, 1994] is an abstraction of an ordinary differential equation, consisting of a set of real-valued variables and functional, algebraic and differential constraints among them, see Appendix A section A.2 for more details. A *QDE* model is qualitative in two senses:

- The values of variables are described in terms of their ordinal relations with a finite set of symbolic landmark values, rather than in terms of real numbers.
- Functional relations may be described as monotonic functions rather than by specifying a functional form.

These purely qualitative descriptions can be augmented with semi-quantitative knowledge in the form of real bounding intervals around unknown real values and real-valued bounding envelope functions around unknown real-

[3] We also tested our algorithms in a similar domain, Chapter 6, section 6.2.

valued functions. Qualitative and semi-quantitative models can be derived by composing model fragments and collecting the associated modelling assumptions.

Qualitative Simulation starts with a *QDE* and a qualitative description of an initial state. Given a qualitative description of a state, it predicts the qualitative state descriptions that can possibly be direct successors of the current state description. Repeating this process produces a graph of qualitative state descriptions, in which the paths starting from the root are the possible qualitative behaviours. The graph of qualitative states is pruned according to criteria derived from the theory of ordinary differential equations, in order to preserve the guarantee that all possible behaviours are predicted. Abstraction methods have also been developed to simplify the resulting qualitative behaviours.

The resulting graph of qualitative states can still be quite large, requiring automated methods based on temporal logic model-checking to determine whether the qualitative prediction implies a desired conclusion. A set of qualitative models and their associated predictions can also be unified with a stream of observations to monitor an ongoing dynamical system or to do system identification on a partial model.

There has been much work on Qualitative Simulation for dynamic systems: [Berleant and Kuipers, 1992] [Dvorak and Kuipers, 1989; Dvorak, 1992; Berleant, 1991; Berleant and Kuipers, 1997; Clancy, 1998; Clancy, 1997; Clancy and Kuipers, 1998; Kay *et al.*, 2000; Rinner and Kuipers, 1999b]. Other important works in Qualitative Simulation are [Subramanian, 1995; Kay and Ungar, 2000; Molle, 1989; Molle and Edgar, 1990].

When complete information about an industrial process is not available, quantitative model-based techniques for *FDI* can be replaced by qualitative ones. These make use of the available incomplete information by building a qualitative model, in terms of which the analysis and reasoning can be carried out.

A qualitative observer makes use of Qualitative Simulation on the basis of conventional filtering techniques to perform observation filtering. The principle of observation filtering is that the simulated qualitative behaviour of a variable must cover its counterpart, the measurement obtained from the system itself. The idea behind the qualitative observer-based technique is that a fault causes a deviation of the system output in such a way that its counterpart, the estimated output, is no longer consistent, i.e. a fault will produce an empty set of qualitative estimated states [Zhuang and Frank, 1997].

[McIlraith *et al.*, 1999] formulated the diagnosis problem for dynamical systems as a model selection problem, where hybrid systems were considered. They divide the diagnosis task into two steps, initial conjecturing of candidate diagnoses followed by subsequent refinement and tracking to select the most likely diagnoses. This application combines different qualitative reasoning techniques such as temporal causal graphs [Mosterman and Biswas, 1999] and trackers [Rinner and Kuipers, 1999a] with quantitative approaches like the Expectation-Maximization method [Dempster *et al.*, 1977] and the General Likelihood Ratio, [Basseville and Nikiforov, 1993].

[Lafortune *et al.*, 2001; Sampath, 2001] proposed an approach based on Discrete Event Systems[4] (*DES*) to perform detection and identification of unobservable fault events using diagnosers. These are finite-state automata built from the discrete-event model of the system under consideration. This hybrid approach to failure diagnosis integrates the qualitative discrete event systems diagnostic methodology with quantitative analysis-based techniques.

2.5.8 Logic-based approach

Within the reasoning methods, the logic-based approach considers the knowledge of the domain to be defined through first-order clauses. These clauses define the design (functionality and structure) of the process, the possible

[4]DESs are dynamic systems which are characterized by a discrete state space of logical values and by event-driven dynamics.

faults and the observations that the process can generate. There are two main logic-based approaches [Poole, 1989]: Consistency-based and Abductive-based.

- *Consistency-based approach.*
  In this approach there is only knowledge about how the process operates normally, how its parts are structured. The diagnosis task consists of isolating deviations from normal performance. There is no knowledge about how the faults occur and appear themselves [Reiter, 1987]. A consistency-based diagnosis is a minimal set of abnormalities such that the observations are consistent with all other components under normal performance. The main idea behind consistency-based diagnosis is that there always exist representative observations when the process works normally.

  [Reiter, 1987] presents a consistency-based diagnosis approach which can be applied to dynamic processes, but it demands checking the consistency of process behaviour over time. Tracking the actual behaviour over time, and simulation, are required. This simple idea was used in many applications. See [Dvorak, 1992] as an example, in which qualitative reasoning becomes necessary in order to cope with the prohibitive solution space. [Struss, 1997] gives a theoretical foundation for consistency-based diagnosis without the expensive simulation step.

- *Abductive-based approach.*
  Here there is only knowledge about the faults and how they appear. An abductive diagnosis hypothesizes faults in order to explain the observed symptoms. An abductive diagnosis is a minimal set of assumptions that, along with background knowledge, implies the observation.

  [Poole, 1987] describes the Theorist framework, an example of how to use abductive-based diagnosis. Given some observations, the system builds the theory that would explain these observations. The Theorist system is implemented with a theorem prover. [de Silva *et al.*, 1992] developed a hybrid diagnostic system based on the Theorist framework and artificial neural networks. Abductive diagnosis provides mechanisms that permit more detailed diagnosis than consistency-based diagnosis can provide.

*Independent Choice Logic (*ICL*)* [Poole, 1997] combines the power of expressibility of first-order logic with a structured approach to handle uncertainty with probabilities. The *ICL* can represent both types of logic-based approaches, consistency-based and abductive-based. Moreover, *ICL* combines logic, decision and game theory into a coherent framework. It has a simple possible-worlds semantics characterized by independent choices and an acyclic logic program that specifies the consequences of these choices.

Probabilistic Horn Abduction (*PHA*) [Poole, 1993] is an extension of *ICL* which includes acyclic logic programs with Negation as Failure [Poole, 2000]. *ICL* has a structured hypotheses space, and an acyclic logic program to give the consequences of the hypotheses. The hypotheses are partitioned into alternatives, the set of alternatives is a choice space. There is a possible world for each selection of one element from each alternative; the logic program specifies what is true in that world. The semantics of negation as failure is given in terms of stable models.

Another extension of *ICL* is dynamic independent choice logic, where the dynamics of the world is modelled rather than the structure of the choices. One can represent any discrete Bayesian network with first-order clauses. This feature can be exploited to model cause-effect relationships for the fault diagnosis task. Computing the posterior probabilities will help to determine the most likely set of faulty components in the process. The probable faulty components are generated by computing the most likely explanations.

[Garza *et al.*, 2001] proposed a methodology for automated diagnosis where a Dynamic *ICL* is used to represent the diagnosis problem. Causal probabilistic models are used to represent the relationships among the elements of

the process and its dynamics. Additionally, *FDI* techniques are integrated into the probabilistic logic framework.

2.5.9 Dynamic Bayesian Network approach

A Dynamic Bayesian Network (*DBN*) [Dean and Kanazawa, 1989] is a temporal stochastic model for dynamic systems which is suitable for monitoring processes. Bayesian Networks (*BN*) are directed graphical models that encode conditional independences between state variables via the graphical structure. State variables are represented as nodes and influences between variables are represented as arcs between nodes. Each node is associated with a given conditional probability distribution that encapsulates the conditional probability of that variable given its parents in the network.

A dynamic Bayesian network is a temporal version of a *BN*, in which nodes represent the state of the system at a particular point in time. The set of nodes representing the state at a point in time is called a time slice. Nodes in one time slice may have parents in the same time slice and in the previous time slice. A *DBN* is specified by two components: a prior *BN* that represents the initial distribution over the initial state, and a 2-time-slice *BN* that represents the transition distribution from states at time t to states at time $t+1$.

[Lerner *et al.*, 2000] proposed an approach to the problem of tracking and diagnosing processes with mixtures of discrete and continuous variables. The approach is based on the framework of hybrid dynamic Bayesian networks. Hybrid *DBN*s can manage a greater range of problems such as nonlinear dynamics and discrete failure modes that influence system evolution. Also, hybrid *DBN*s can model a variety of faults, including burst faults, measurement errors and gradual drifts. Many of the issues that have challenged traditional approaches to diagnosis – ranking possible failures, handling of multiple simultaneous failures, and robustness to parameter drift – can be addressed within a probabilistic tracking framework.

The inference task in *DBN*s is generally intractable because the number of modes in these systems grows exponentially over time. [Lerner *et al.*, 2000] proposed an approach based on the same Kalman filter principles. They maintain multiple candidate hypotheses about the state of the process, updating them based upon evidence. Similar hypotheses are collapsed instead of choosing the most likely ones. Using a bound window one step forward into the future, they propose using this information to determine which hypotheses should be kept and which should be collapsed. They avoid the exponential blowup, caused by many discrete variables, by reasoning separately about the different subsystems while still propagating correlations between them. A simulated faulty process is analyzed.

[Lerner *et al.*, 2002] demonstrated the feasibility of the hybrid *DBN* approach for monitoring complex processes. A general framework is given for approximating nonlinear behaviour using integration methods that extended the *unscented filter* [Julier and Uhlmann, 1997]. Also, they show how to use a fixed-point computation to deal with effects that develop at different time scales. Experimental results indicate that this approach is much faster and more precise than standard particle filtering.

[Arroyo-Figueroa and Sucar, 1999] presented a type of probabilistic temporal network called temporal nodes Bayesian network (*TNBN*). These networks combine uncertainty and temporal reasoning. A *TNBN* is a Bayesian network where each node represents an event change of a variable, and causal-temporal relationships are included through the arcs. Previous probabilistic temporal models[5] represent the state value at different times, however *TNBN*'s representation is based on state changes at different times. This representation was tested for diagnosis using a steam boiler training simulator. The level steam drum behaviour was analyzed after an increment on steam demand occurs. This is a well-known challenging problem for boiler's operation, specially, if the *shrink-swell* phenomena appears, [Dukelow, 1991]. If there are few changes for each variable in the time interval of inference,

[5] See related work in the extended version paper, [Arroyo-Figueroa *et al.*, 1998].

the modelling task becomes easy. Moreover, this facilitates the temporal knowledge acquisition (one of the most time consuming task). A practical application of these ideas were consolidated through the SEDRET[6] system, [Arroyo-Figueroa *et al.*, 2000]. Potentially, SEDRET can be used to assist an operator in real-time operation for a steam boiler.

2.6 Particle Filtering in diagnosis

Particle Filters (*PF*) are powerful tools for Bayesian state estimation in non-linear systems, [Doucet *et al.*, 2001; Doucet, 1998; Pitt and Shephard, 1999; Kanazawa *et al.*, 1995; Liu and Chen, 1998]. One can approximate a posterior distribution over unknown state variables using a set of particles drawn from this distribution.

[Thrun *et al.*, 2002] proposed a Particle filter that generates samples according to a distribution and combines the posterior probability with a *risk function*. The risk function measures the importance of a state location on future cumulative costs. This approach yields better results than conventional Particle filtering for fault diagnosis. The algorithm was tested in two domains: robot localization and mobile robot diagnostics. [Thrun, 2002] presents some of the recent innovations in robotics using *PF*. Early successes for Particle filter implementations can be found in the area of robot localization, in which a robot's position must be recovered from sensor data. Particle filters were able to solve two important previously unsolved problems, known as the global localization and kidnapped robot problems, in which a robot has to recover its position under global uncertainty.

[Verma *et al.*, 2001] gave a method for autonomous fault detection in simulated space rovers. The approach uses a non-parametric estimate of system state which is updated based on sensor measurements. The state estimation is generated using a decision-theoretic generalization of Particle filters which uses utility functions to detect the important states. Good results were shown in terms of variance and number of particles in a small domain. However, the algorithm is computationally very expensive.

[Dearden and Clancy, 2001] described an approach to hybrid diagnosis based on Particle filters for planetary rovers. The hybrid model can be seen as a partially observable Markov decision process (POMDP) [Monahan, 1982]. POMDP is a framework for decision-theoretic planning problems, where the task is to determine the best action to perform given the current estimate of the actual state of the system. This estimate, commonly named the belief state, is what they want to determine in the diagnosis problem. Maintaining an exact belief state is computationally intractable for this kind of domain. Therefore, they approximate the belief state and keep it updated using Particle Filtering. However, because fault states typically have very low probability, there is a risk that there will be no particles in a fault state when a fault occurs, and the system will be unable to diagnose the fault. The proposed solution always tries to assign some particles to fault states that are important to diagnose. This is done without biasing the diagnosis. These important states are suggested by a traditional model-based diagnosis system.

[de Freitas, 2001] proposed an efficient Monte Carlo method known as *Rao-Blackwellized* Particle Filtering (RBPF) [Doucet *et al.*, 2000a] for fault diagnosis. The task of diagnosis is to identify the discrete state of operation using continuous measurements. A jump Markov linear Gaussian model was adopted, consisting of one different linear-Gaussian state model for each possible discrete state. The results show a considerable reduction in diagnosis error.

[Verma *et al.*, 2002] presents an integration of their early, [Thrun *et al.*, 2002; Verma *et al.*, 2001], approaches, but they include probabilistic methods to account for the uncertainty. They present a new hierarchical approach that improves the accuracy of fault identification by focusing the search for the correct fault hypothesis. The algorithms

[6] Intelligent System for the Diagnosis and Prediction of Events

were tested using the Darwin2K simulator [Leger, 2000].

[Koller and Lerner, 2001] tested a Particle Filtering approach. *Bootstrap* sampling algorithm [Elliott, 1994] (with some modifications in the implementation) is applied to two large *DBN*s. One problem is a hybrid network that models a nonlinear process (two water tanks) which is commonly used as a benchmark in the fault diagnostics community [Mosterman and Biswas, 1999]. They show that particle filtering works properly with an extremely rich class of dynamic systems. Particle filtering showed robustness and high applicability to different domains; however, the curse of dimensionality is an issue for Particle Filtering in high-dimensional spaces. The computing time required was not discussed.

[McIlraith, 2000] builds on a previous work, [McIlraith *et al.*, 1999] (discussed in subsection 2.5.7). It presents a mathematical formulation of the hybrid monitoring and diagnosis task as Bayesian model tracking and model selection. Basically, they give a method for tracking multiple models of nonlinear behaviour simultaneously using factored sampling and conditional density propagation. The condensation [Isard and Blake, 1998] Particle Filtering algorithm is used.

[Ng *et al.*, 2002] proposes an approximate monitoring algorithm that combines the best qualities of Particle Filtering and the Boyen-Koller (*BK*) method [Boyen and Koller, 1998]. Like *PF*, it can be performed on any size model, and like BK, it is able to exploit the subdivision of a complex system into weakly interacting subsystems. The basic idea is to represent the belief state using a set of factored particles. By factoring particles, a bias is introduced into the belief approximation, however, the variance is significantly reduced. Three implementations of the Factored Particles algorithm were created using dynamic Bayesian networks with binary nodes.

[Koutsoukos *et al.*, 2002] proposed a hybrid diagnosis architecture based on Particle Filtering. The approach combines Qualitative Reasoning and hybrid estimation, leveraging the speed of the Qualitative diagnoser and the resolution of the hybrid model. Basically, the approach uses continuous measurements to compute appropriate likelihood functions, but it is based on a temporal discrete event model of the system dynamics. Autonomous transitions between modes triggered by the continuous dynamics are considered. This hybrid diagnosis is focused on the mode transitions that cover most of the probability space. Probabilities change dynamically based on the continuous behaviour of the system and have to be recomputed at every time step. Particle filtering allows an efficient computation of these transition probabilities. The proposal was only tested with simulated processes.

2.7 Conclusions

We consider the inference task (diagnosis/estimation) to be the determination of the state of the continuous dynamic process over time given some observations, [de Freitas, 2001]. We want to reason with a hybrid model that represents the continuous dynamic system behaviour. This hybrid model contains continuous and discrete variables. A set of discrete modes represents the different states (faulty or non-faulty operating conditions) of the system, and a set of continuous values represents the continuous variables that govern the dynamics of the system.

Computing exact inference for a hybrid dynamic model such as the one we describe above is computationally intractable. The complexity of the inference task grows exponentially over time, so no closed-form solution is possible. Approximate inference algorithms such as Particle Filtering are the most general approach to this problem, and currently the only approach to performing filtering[7] in general-purpose hybrid dynamic Bayesian networks.

It is important to insist that we model the continuous dynamic system using a hybrid dynamic Bayesian network, where discrete nodes represent the discrete modes and continuous nodes represent the continuous observations.

[7] Also called tracking or monitoring.

However, we do not adopt any conventional approximate inference techniques for this representation, because these techniques are designed primarily for discrete domains or very limited class hybrid *DBN*s.

Other features that we want in our solution are:

- Nonlinearity in processes. Non-linear processes can be qualitatively linearized around some specific operating points.
- Low probabilities. Faulty states typically have very low probabilities; this causes problems when numerical approximations are applied.
- On-line solutions. We would like to be able to do inference in real-time, although off-line solutions are quite acceptable in many domains.

Only Particle Filtering techniques can cope with these requirements. Therefore, we propose a principled probabilistic approach to the on-line diagnosis/estimation of dynamic systems.

Chapter 3

Fundamentals

3.1 Introduction

A Particle Filter (*PF*) is a Markov chain Monte Carlo (*MCMC*) algorithm that approximates the belief state using a set of samples, called particles, and keeps the distribution updated as new observations are made over time. Rao-Blackwellized Particle Filtering (*RBPF*) is a Particle Filtering variant. Both Particle Filtering and *RBPF* have been successfully used in dynamic systems for inference tasks.

Typical dynamic industrial processes have multiple operating conditions, called discrete modes. If each discrete mode's behaviour has linear dynamics, we can model a process with the Jump Markov Linear Gaussian (*JMLG*) model.

In this chapter we describe the *PF* and *RBPF* algorithms for dynamic systems which are modelled by the (*JMLG*) model.

First, we describe how the *JMLG* model emerges from the State-Space Model (*SSM*) and the Hidden Markov Model (*HHM*). Then, three learning algorithms be discussed:

- *Variational learning*, which maximizes a lower bound on the log likelihood.
- *Expectation-Maximization*, an iterative procedure for maximum likelihood parameter estimation from data sets with missing or hidden variables.
- *Least Squares Estimation*, which aims to minimize the sum of the squared deviations of the observed values for the dependent variable from those predicted by the model.

These algorithms are combined in the following chapter in order to find a learning procedure for the *JMLG* parameters.

Next, we present in detail the Particle Filtering and Rao-Blackwellized Particle Filtering algorithms. The sequential importance sampling and selection steps (basic steps in *PF*) are discussed. Also, the Kalman filter is analyzed as an important step in the *RBPF* algorithm.

Finally, we summarize this chapter, which includes the most important fundamentals on which our research is based.

3.2 Jump Markov Linear Gaussian Systems

Many probabilistic time series models come from either Hidden Markov Models (HMMs) or stochastic linear dynamical systems commonly known as State-Space Models (SSMs). Using a single discrete random variable – the *hidden* state – hidden Markov models can represent the past information of a sequence. The prior probability distribution of this state can be calculated from the previous hidden state using a stochastic transition matrix. If we know the state at any time, the past, present and future observations become statistically independent – the *Markov independence property.*

Similarly, using a real-valued hidden state vector, state-space models can represent past information. Again, conditioned on this state vector, the past, present, and future observations are statistically independent. The dependency between the present state vector and the previous state vector is specified through the dynamic equations of the system and the noise model. A common case occurs when these equations are linear and the noise model is Gaussian; this model is also known as a linear dynamical system or Kalman filter model. HMMs and SSMs are well-known models; however, most real and interesting processes cannot be characterized by either purely discrete or purely linear-Gaussian dynamics.

Typical industrial processes, such as the one we described in Chapter 2 subsection 2.3.1, may have multiple discrete modes of behaviour, each of which has approximately linear dynamics. We are interested in dynamical systems which are characterized by a combination of discrete and continuous dynamics. Switching state-space models, or Jump Markov Linear Gaussian (JLMG) systems, are a natural generalization of hidden Markov models and state-space models in which the dynamics can transition in a discrete manner from one linear operating regime to another. This model had been exploited in many fields including digital communication [Logothetis and Krishnamurthy, 1999], signal processing [Krishnamurthy and Moore, 1993], target tracking [Bar-Shalom and Li, 1995; Mazor *et al.*, 1998], economics, and many other fields [Chang and Athans, 1978; Shumway and Stoffer, 1991; Bar-Shalom and Li, 1993]. In the following subsections, we briefly present the *SSM* and *HMM* frameworks and then the *JMLG* model.

3.2.1 State-Space Model (SSM)

State-space models are commonly used in signal processing. A state-space model defines a probability density over a time series of real-valued observation vectors by assuming that the observations were generated from a sequence of hidden state vectors. The hidden state vectors obey the Markov independence property. The joint probability for the sequences of states and observations can be represented as[1]:

$$p(x_{1:T}, y_{1:T}) = p(x_1)p(y_1|x_1)\prod_{t=2}^{T} p(x_t|x_{t-1})p(y_t|x_t) \tag{3.1}$$

Figure 3.1 shows the conditional independences specified by equation (3.1). Figure 3.1 is a Directed Acyclic Graph (DAG). Each node is conditionally independent of its non-descendents given its parents. Specifically, the variable y_t is conditionally independent of all other variables given the state x_t; and x_t is conditionally independent of $x_1, ..., x_{t-2}$ given x_{t-1}. Shaded nodes represent observable variables and unshaded nodes represent hidden variables.

The simplest model assumes that the transition and output functions are linear and time-invariant, and that the distributions of the state and observation variables are multivariate Gaussian. The state transition function is

$$x_{t+1} = Ax_t + B\gamma_{t+1} \tag{3.2}$$

[1] See Appendix B, Section B.1.1 for notation.

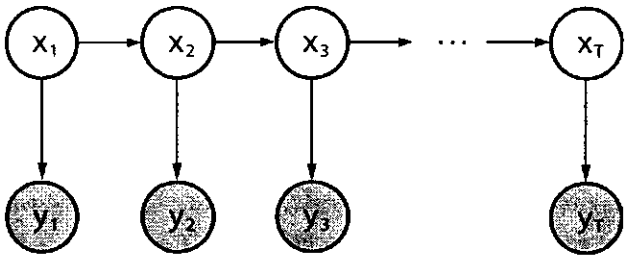

Figure 3.1: State-Space Model graph. For this directed acyclic graph, each node is conditionally independent of its non-descendents given its parents. Shaded nodes represent observed variables and unshaded nodes represent hidden variables. An initial state $x_0 \sim \mathcal{N}(\mu_0, \sum_0)$ is assumed. The Gaussian noises are omitted for clarity.

where A is a state transition matrix, B is the noise state matrix, γ_t is $i.i.d.$ Gaussian, such as, $\gamma_t \sim \mathcal{N}(0, I)$ with covariance $\mathcal{Q}$. The initial state is $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$. Equation (3.2) ensures that if $p(x_t)$ is Gaussian, then $p(x_{t+1})$ is Gaussian too. The output function is

$$y_t = Cx_t + Dv_t \tag{3.3}$$

where C is the output matrix, D is the output noise matrix, v_t is $i.i.d.$ Gaussian, such as $v_t \sim \mathcal{N}(0, I)$ with covariance $\mathcal{R}$. Often, the observation vector can be divided into input (u_t) and output (y_t) variables. Equation (3.2) becomes

$$x_{t+1} = Ax_t + B\gamma_{t+1} + Fu_{t+1} \tag{3.4}$$

where u_t is the input observation and F is the input matrix. Equation (3.4) models the input-output behaviour. The general output function is

$$y_t = Cx_t + Dv_t + Gu_t \tag{3.5}$$

where G is usually a null matrix for most applications. Figure 3.2 shows a full state-space representation including input variables.

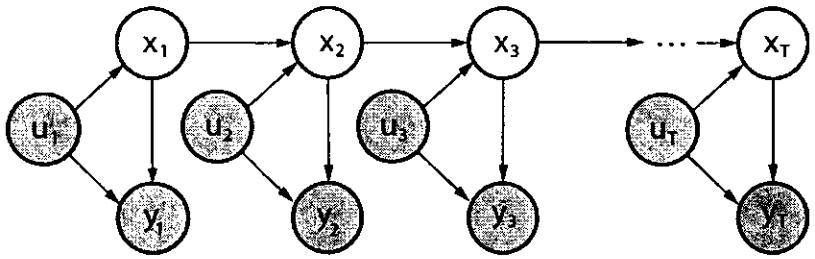

Figure 3.2: Full state-space model. A probabilistic graphical model for stochastic dynamic systems with hidden states (x_t), and observable (inputs u_t and outputs y_t) variables. The output at time t is conditionally independent of all other variables given the state at time t. The state at time t is conditionally independent of states at times $1, 2, \ldots, t-2$ given the previous state at $t-1$. Gaussian noises (w_t, v_t) are omitted for clarity.

$p(y_t|x_t)$ is also Gaussian, given by equation (3.6)

$$p(y_t|x_t) = (2\pi)^{-\frac{n_y}{2}}|R|^{-\frac{1}{2}}exp\left[-\frac{1}{2}(y_t - Cx_t - Gu_t)'\mathcal{R}^{-1}(y_t - Cx_t - Gu_t)\right], \tag{3.6}$$

The problem of *state estimation* or *inference* for state space models consists of estimating the posterior probabilities of the hidden variables given a sequence of the observed variables. Assuming the local likelihood functions for the observations are Gaussian and the priors for the hidden states are Gaussian, the resulting posterior is also Gaussian. The most important special cases of inference are:

- *Filtering* (also called monitoring or tracking), where the goal is to calculate the probability of the current hidden state x_t give the sequence of inputs and outputs up to time t, $p(x_t|y_{1:t}, u_{1:t})$. The Kalman Filter (KF) is the solution to this problem.
- *Smoothing*, in which the goal is to calculate the probability of x_{t_s} given the sequence of inputs and outputs up to time t, where $t_s < t$, $p(x_{t_s}|y_{1:t}, u_{1:t})$ is computed using the Kalman filter in its forward direction. A similar set of backward recursions from t_s to t completes the computation by accounting for the observations after time t. This idea is known both as the Kalman smoother and the RTS (Rauch-Tung-Streibel) smoother [Rauch *et al.*, 1965].
- *Prediction*, where the goal is to calculate the probability of future states and observations given observations up to time t. We can simulate the system in the forward direction given $p(x_{t_p}|y_{1:t}, u_{1:t})$ and compute the probability density at future time t_p, $t_p > t$.

A graphical representation of these kinds of inference is shown in Figure 3.3.

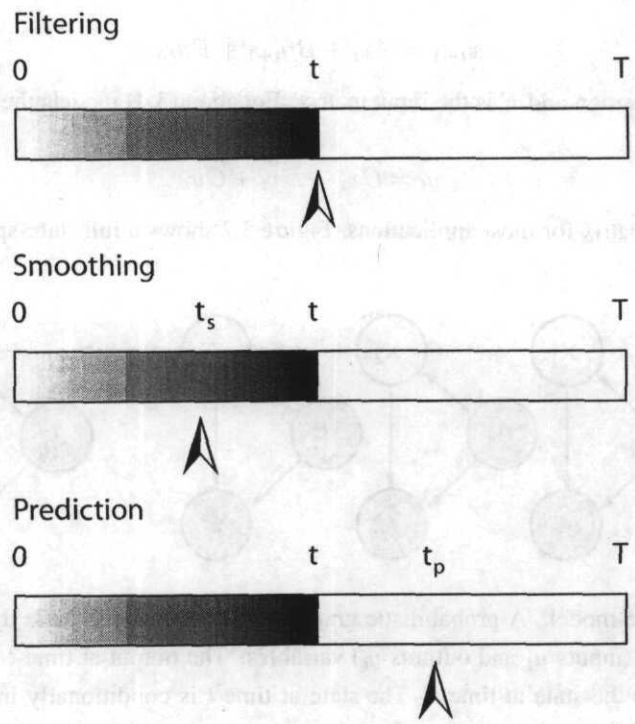

Figure 3.3: Three kinds of inference. Inference is computed by $p(x_t|y_{1:t}, u_{1:t})$. We know the data represented by the shaded area. t is the current time. T is the maximum time. The triangle indicates the time step (t for Filtering, t_s for Smoothing, and t_p for Prediction) in which the inference is computed.

3.2.2 Hidden Markov Model (HMM)

Like the state space model, the hidden Markov model defines probability distributions over sequences of observations, $y_{1:t}$. The distribution over sequences is obtained by specifying a distribution over observations at each time step t given a discrete hidden state z_t (as opposed to the continuous state in an SSM), together with the probability of transitioning from one hidden state to another. The joint probability for the sequences of states z_t and observations y_t can be factored as in equation (3.7)

$$p(z_{1:T}, y_{1:T}) = p(z_1)p(y_1|z_1)\prod_{t=2}^{T} p(z_t|z_{t-1})p(y_t|z_t) \tag{3.7}$$

This equation obeys the Markov independence property. Figure 3.4 shows the conditional independencies specified by equation (3.7), where $z_0 \sim p(z_0)$.

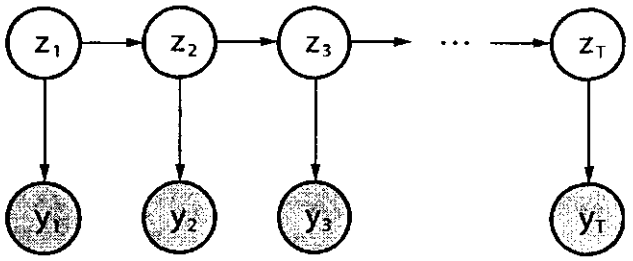

Figure 3.4: Hidden Markov model graph. Conditional independencies are represented as in the state space model; however, z_t represents a *discrete* hidden state.

In the HMM framework, the state is represented by a single multinomial variable; this variable can take one of n_z discrete values, $z_t \in \{1, \ldots, n_z\}$. The state transition probabilities are defined by $p(z_t|z_{t-1})$. If the observables are discrete symbols taking one of n_y values, the observation probabilities is represented by $p(y_t|z_t)$. However, for a continuous observation vector, $p(y_t|z_t)$ can be modelled in many different forms, such as Gaussian, a mixture of Gaussians, etc.

Given sequences of observations, we can use the *Baum-Welch* algorithm [Baum *et al.*, 1970] and learn maximum likelihood parameters for an HMM. The *Baum-Welch* algorithm follows *Expectation-Maximization* (EM) principles, [Dempster *et al.*, 1977]. For the E-step, it uses the forward-backward algorithm to infer the posterior probabilities of the hidden states; for the M-step, it uses expected counts of transitions and observations to re-estimate the transition and output matrices.

The inference task in hidden Markov models can be addressed using two algorithms. Given a hidden Markov model with known parameters and a sequence of observations, we can compute the posterior probabilities of the hidden states using a recursive algorithm known as the *forward-backward* algorithm. The *forward-backward* algorithm follows the Kalman filter and Kalman smoother principles. If we want to compute the single most likely sequence of hidden states, the *Viterbi* algorithm is the solution. This algorithm also consists of a forward and backward pass through the model. Like state space models, Figure 3.2, hidden Markov models can be augmented to allow for input variables [Bengio and Frasconi, 1995].

Hidden Markov models and State Space models are well-known frameworks with many more extensions and algorithms to work with. We combine these simple structures in order to get a hybrid model which is a rich enough representation for our purposes.

3.2.3 Hybrid models

A natural way to improve both models is to combine them. Such combinations are known as hybrid models, state-space models with switching, and jump-linear systems. Basically, hybrid models combine the discrete transition structure of hidden Markov models with the linear dynamics of state space models. A lot of work has been done using this idea in different domains.

Inference approaches in switching state space models

One of the very first approaches to addressing the inference task in this kind of hybrid model was developed by [Chang and Athans, 1978]. They computed the conditional mean and variance of the state for linear switching state-space models. However, the prior and transition probabilities of the switching process are assumed to be known. They show that the exact conditional distribution of the state is a Gaussian mixture with $(n_z)^T$ components.

[Bar-Shalom and Li, 1993] addressed the state-estimation problem in switching models using different methods. These methods are referred to as generalized pseudo-Bayesian and Interacting Multiple Models. They follow the same idea of collapsing the mixture of n_z Gaussians (which results from considering all the settings of the switch state at a given time step) into a single Gaussian. This approximated solution avoids the exponential growth of mixture components. Figure 3.5 shows the hybrid model considered. Both the state dynamics and the output matrices switch, and the switching obeys Markovian dynamics.

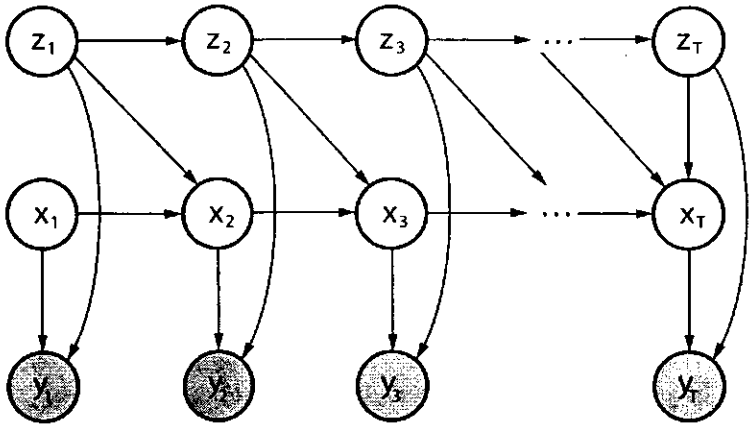

Figure 3.5: Hybrid model. State and output matrices switch at each time step, obeying the Markovian property.

[Kim, 1994] derived approximation methods for similar hybrid models, adding observable inputs, Figure 3.6. The James Hamilton Markov switching model is extended to the state space representation of a general dynamic linear model, which includes autoregressive integrated moving average and classical regression models as special cases.

[Hamilton, 1994] proposed a hybrid model in which a real-valued observation at time t, y_t is Gaussian. Its mean is a linear function of $y_{t-1}, ..., y_{t-r}$ and of binary indicator variables for the discrete states, $z_{t-1}, ..., z_{t-r}$. Basically, the system is an $(r+1)^{th}$ order hidden Markov model driving an r_{th} order auto-regressive model. This system is tractable for small r and a small number of discrete states n_z. The real-valued states are known (observable). Outputs depend on the states and previous outputs, and the form of this dependence can switch randomly, Figure 3.7. This hybrid model is closely related to the Hidden Filter HMM (HFHMM), [Fraser and Dimitriadis, 1993]. Exact

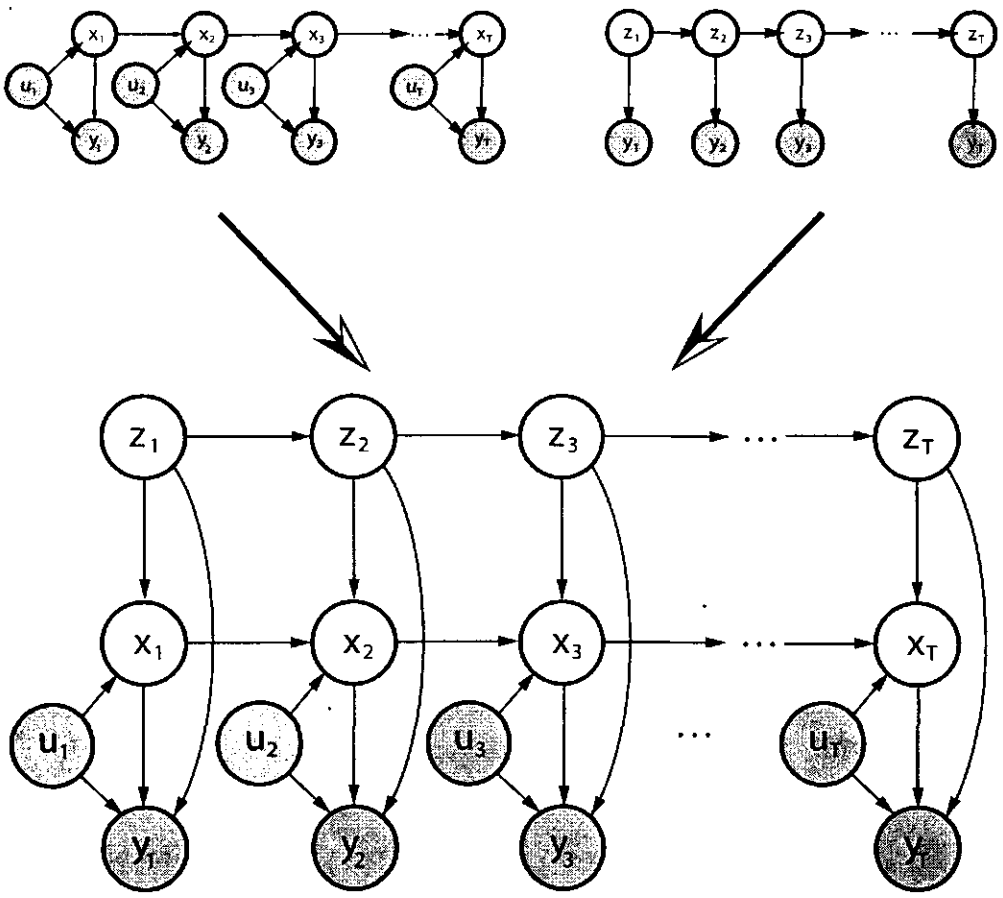

Figure 3.6: Hybrid model with observable inputs. State and output matrices switch at each time step. This hybrid model combine both the discrete transition structure hidden Markov models with the linear dynamics of state space models.

inference in this model can be carried out tractably, using an algorithm which works like the forward-backward procedure for hidden Markov models.

An inference algorithm for hybrid Markov switching systems was presented by [Elliot *et al.*, 1995]. The true switch states, z_t, are represented as unit vectors, and the estimated switch state is a vector in the unit square with elements corresponding to the estimated probability of being in each switch state. The real-valued state, x_t, is approximated as a Gaussian given the estimated switch state, formed from a linear combination of the transition and observation matrices for the different SSMs weighted by the estimated switch state.

3.2.4 JMLG model

We work with the following hybrid model, the *Jump Markov Linear Gaussian* (*JMLG*) model, Figure 3.6. The dynamic behaviour for the simplest case of this model was described by equations (3.4-3.5). We generalize it here:

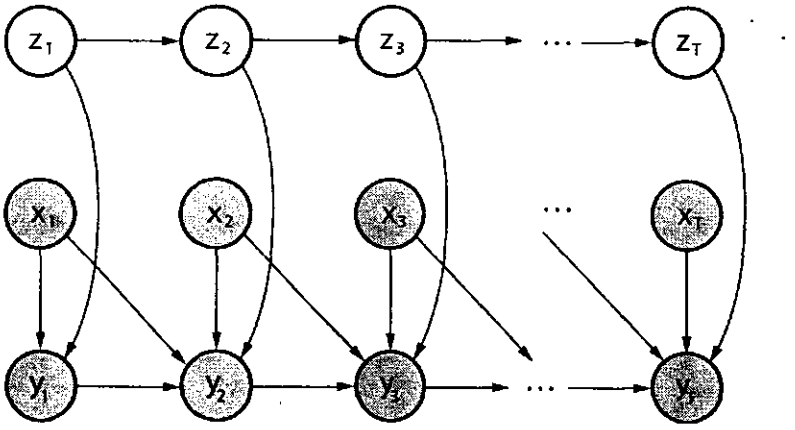

Figure 3.7: Hybrid model; states and outputs are observable.

$$z_t \sim p(z_t|z_{t-1}) \tag{3.8}$$

$$x_t = A(z_t)x_{t-1} + B(z_t)\gamma_t + F(z_t)u_t \tag{3.9}$$

$$y_t = C(z_t)x_t + D(z_t)v_t + G(z_t)u_t, \tag{3.10}$$

where $y_t \in \mathbb{R}^{n_y}$ denotes the measurements, $x_t \in \mathbb{R}^{n_x}$ denotes the unknown continuous states, $u_t \in \mathcal{U}$ is a known input, and $z_t \in \{1, \ldots, n_z\}$ denotes the unknown discrete modes. The noises are *i.i.d.* Gaussian: $\gamma_t \sim \mathcal{N}(0, I)$ and $v_t \sim \mathcal{N}(0, I)$. Note that the parameters $\left(A(i), B(i), C(i), D(i), F(i), G(i)\right)_{i=1}^{n_z}$ depend on the discrete mode. For each discrete mode, we have a single linear-Gaussian model. We ensure that $D(i)D(i)^{\mathrm{T}} > 0$ for any i. The initial states are $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ and $z_0 \sim p(z_0)$.

The above mathematical description for the *JMLG* model is characterized and tested in subsequent chapters.

3.3 Learning the JMLG parameters

3.3.1 Introduction

The problem of learning the parameters of a switching state-space model has not yet been solved. However, important contributions have been made. The most relevant approaches are:

- *Variational Learning* for switching state space models, [Ghahramani and Hinton, 1998].
- The *Expectation-Maximization* method for state space modes (switching state space models with $n_z = 1$), [Ghahramani and Hinton, 1996].

Although the results obtained by these approaches are not totally satisfactory, they are the building blocks for our proposal, particularly the *EM* method. Traditional *Least Squares Estimation* is also part of our proposal, and is also reviewed.

[Tugnait, 1982] used the truncated maximum likelihood estimation algorithm to estimate the transition probability given the noisy observations of the system output variables. Switchings are modelled by a finite state ergodic

Markov chain. However, the transition probability matrix is assumed to belong to a compact set. The measurements are taken after the system has achieved a statistical steady state – another important limitation.

The learning problem for switching state-space models was addressed by [Shumway and Stoffer, 1991]. They consider a single real-valued hidden state vector and switching output matrices. The probability of choosing a particular output matrix is a pre-specified time-varying function (independent of previous choices). The approach involves classical maximum likelihood estimation using an approximation to the *EM* algorithm in combination with a standard nonlinear optimization procedure. Figure 3.8 shows the kind of hybrid model considered. Their algorithm uses a single Gaussian to approximate a Gaussian mixture with $(n_z)^T$ components at each time step.

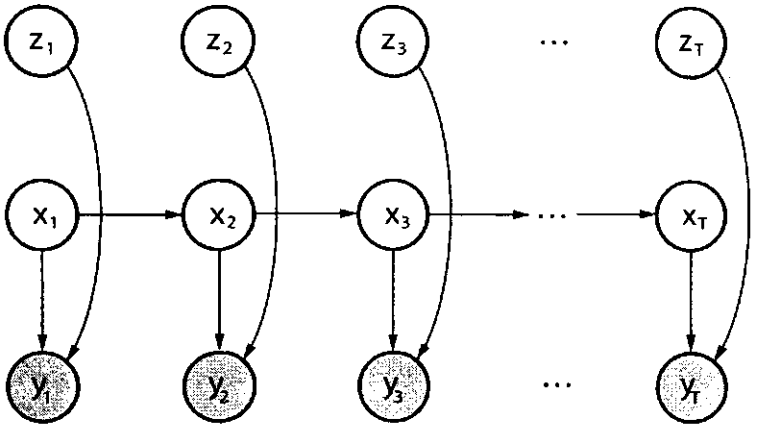

Figure 3.8: Hybrid model. The output matrix switches independently at each time step.

3.3.2 Variational Learning

[Ghahramani and Hinton, 1998] proposed a hybrid model: a state-space model with switching and a jump linear system with hidden multiple real-valued state vectors, Figure 3.9. The jump Markov linear Gaussian model that we want to exploit has the architecture shown here, Figure 3.9. It is important to emphasize that this graphical representation correspond to the same model represented in Figure 3.6. For simplicity, in Figure 3.9 we omitted the input signal u_t and the relationship between z_t and $\{x_t^j\}_{j=1}^{n_z}$ is considered through the superscript. The proposed inference algorithm is based on a structured variational approximation. A learning algorithm is developed for all the parameters of the model, including the Markov switching parameters. This algorithm maximizes a lower bound on the log likelihood of the data. Basically, it decouples into forward-backward recursions on a hidden Markov model, plus Kalman smoothing recursions on each state space model. The states of the hidden Markov model determine the soft assignment of each observation to a state space model; the prediction errors of the state space model determine the observation probabilities for the hidden Markov model..

The sequence of observations $y_{1:T}$ is modelled by specifying a probabilistic relation between the observations and the hidden state space, which consists of real-valued state vectors $\{x_t^m\}_{m=1}^{n_z}$ and one discrete state space vector z_t. The discrete state, z_t, is modelled as a multinomial variable that can take on n_z values: $z_t \in \{1, \ldots, n_z\}$. The

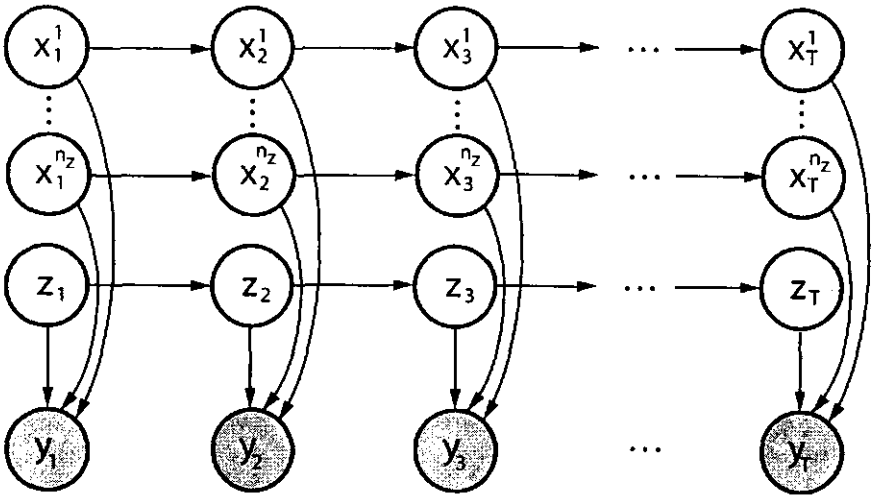

Figure 3.9: Switching state space model. z_t are the discrete switch variables, x_t^m are the real-valued state vectors where $m \in \{1, \ldots, n_z\}$, and y_t are the observable variables.

joint probability of observations and hidden states can be factored as

$$\begin{aligned} p(z_{1:T}, x_{1:T}^{(1)}, \ldots, x_{1:T}^{(n_z)}, y_{1:T}) &= p(z_1) \prod_{t=2}^{T} p(z_t|z_{t-1}) \times \prod_{m=1}^{n_z} p(x_t^{(m)}|x_{t-1}^{(m)}) \\ &\quad \times \prod_{t=1}^{T} p(y_t|x_t^{(1)}, \ldots, x_t^{(n_z)}, z_t), \end{aligned} \tag{3.11}$$

The observable variable y_t is multivariate Gaussian with output equation given by the state space model with switch state $z_t = m$. The probability of the observation vector, y_t, is

$$\begin{aligned} p(y_t|x_t^{(1)}, \ldots, x_t^{(n_z)}, z_t = m) &= (2\pi)^{-\frac{n_y}{2}} |\mathcal{R}|^{-\frac{1}{2}} exp\left[-\frac{1}{2}(\Delta^{(m)} y_t)' |\mathcal{R}|^{-1} (\Delta^{(m)} y_t)\right] \\ where\ \Delta^{(m)} y_t &= y_t - C^{(m)} x_t^{(m)} - G^{(m)} u_t \end{aligned} \tag{3.12}$$

Each real-valued state vector evolves according to the linear Gaussian dynamics of a state space model with its own initial state, transition matrix, input matrix, and noise matrix, see equation (3.4). The switch state variable itself evolves according to the discrete Markov transition structure specified by the initial state probabilities $p(z_0)$ and the state transition matrix $p(z_t|z_{t-1})$.

Using the Expectation-Maximization (EM) algorithm, [Baum *et al.*, 1970; Dempster *et al.*, 1977] (for details see section 3.3.3), a learning algorithm for the parameters of a switching state-space model is proposed. The EM algorithm consists of two iterative steps:

1. The *E-step* optimizes a distribution over the hidden states.
2. The *M-step* optimizes the parameters given the distribution over the hidden states.

Any distribution over the hidden states $q(z_{1:T}, \mathcal{X}_{1:T})$, where $\mathcal{X}_t = [x_t^{(1)}, \ldots, x_t^{(n_z)}]$ is the combined state of the

state space model, can be used to define a lower bound $\mathcal{B}$ on the log probability of the observed data:

$$log\, p(y_{1:T}|\theta) = log \sum_{z_t=1}^{n_z} \int p(z_{1:T}, \mathcal{X}_{1:T}, y_{1:T}|\theta) d(\mathcal{X}_{1:T}) \tag{3.13}$$

$$= log \sum_{z_t=1}^{n_z} \int q(z_{1:T}, \mathcal{X}_{1:T}) \left[\frac{p(z_{1:T}, \mathcal{X}_{1:T}, y_{1:T}|\theta)}{q(z_{1:T}, \mathcal{X}_{1:T})} \right] d(\mathcal{X}_{1:T}) \tag{3.14}$$

θ denotes the parameters of the model. $\theta = \{A^{(m)}, \ldots, G^{(m)}, Q^{(m)}, R^{(m)}, (\mu_o, \Sigma_0), p(z_0), p(z_t|z_{t-1})\}$, where:

- $\{A^{(m)}, \ldots, G^{(m)}\}_{m=1}^{n_z}$ are the matrices of each state space model.
- $Q^{(m)}$ and $R^{(m)}$ are the state and output noise covariances.
- μ_0 and Σ_0 represent the mean and covariance for the initial state x_0.
- $p(z_0)$ is the prior for the initial discrete Markov process.
- $p(z_t|z_{t-1})$ is the discrete transition matrix.

Using Jensen's inequality, [Cover and Thomas, 1991], on equation (3.14), we can get a lower bound $\mathcal{B}(q, \theta)$,:

$$log\, p(y_{1:T}|\theta) \geq \sum_{z_t=1}^{n_z} \int q(z_{1:T}, \mathcal{X}_{1:T}) log \left[\frac{p(z_{1:T}, \mathcal{X}_{1:T}, y_{1:T}|\theta)}{q(z_{1:T}, \mathcal{X}_{1:T})} \right] d(\mathcal{X}_{1:T}) = \mathcal{B}(q, \theta) \tag{3.15}$$

During this iterative procedure, the E-step holds the parameters fixed and sets Q to be the posterior distribution over the hidden states given the parameters, $q(z_{1:T}, \mathcal{X}_{1:T}) = p(z_{1:T}, \mathcal{X}_{1:T}|y_{1:T}, \theta)$. This maximizes $\mathcal{B}$ with respect to the distribution, turning the lower bound into an equality. The M-step holds the distribution fixed and computes the parameters that maximize $\mathcal{B}$ for that distribution. Since $\mathcal{B} = logp(y_{1:T}|\theta)$ at the start of the M-step, and since the E-step does not affect $logP$, the two steps always increase $logP$.

Because the posterior probability of the real-valued states is a Gaussian mixture with n_z^T terms, the exact E-step for a switching SSM is intractable, [Chang and Athans, 1978; Bar-Shalom and Li, 1993]. In order to derive a learning algorithm, [Ghahramani and Hinton, 1998] approximates the posterior probability of the hidden states. Rather than setting $q(z_{1:T}, \mathcal{X}_{1:T}) = p(z_{1:T}, \mathcal{X}_{1:T}|y_{1:T})$ in the E-step, a tractable distribution Q is used to approximate P. The difference between the bound $\mathcal{B}$ and the log likelihood is given by the Kullback-Liebler (KL) divergence between Q and P, [Cover and Thomas, 1991]:

$$KL(q \parallel p) = \sum_{z_t=1}^{n_z} \int q(z_{1:T}, \mathcal{X}_{1:T})\, log \left[\frac{q(z_{1:T}, \mathcal{X}_{1:T})}{p(z_{1:T}, \mathcal{X}_{1:T}|y_{1:T})} \right] d(\mathcal{X}_{1:T}) \tag{3.16}$$

Choosing Q to have a tractable structure, the parameters of Q are varied to minimize equation (3.16). In the new EM algorithm, the E-step optimizes the parameters of the distribution Q to minimize equation (3.16), and the M-step optimizes the parameters of P given the distribution over the hidden states. This strategy is named *variational approximation*[2] and the free parameters of the distribution are called *variational parameters*. This algorithm was tested on artificial data sets, and on a real data set that measured respiration force in a patient with sleep apnea, but the results were not enough good to be a viable method for learning the parameters of switching state-space models[3]. However, this is the most successful general approach in this field, hence this brief explanation.

[2] [Jordan *et al.*, 1999] gives a good introduction to these methods.

[3] Personal communication with Geoffrey Hinton (University of Toronto) and Zoubin Ghahramani(University College London), Dec 2002.

3.3.3 Expectation-Maximization Method

The Expectation-Maximization (*EM*) algorithm, [Dempster *et al.*, 1977], provides a general approach to the problem of maximum likelihood parameter estimation in statistical models (with latent variables). The *EM* approach takes advantage of the conditional independence structure of graphical models. *EM* systematically follows the divide-and-conquer strategy. It helps us deal with problems in which the likelihood or its derivatives are computationally intractable.

Latent variables are generally used to simplify the model. We may observe a complex pattern of dependency among a set of variables $y_{1:T}$. Rather than modelling this dependency directly, we may find it simpler to account for their dependency via "top-down" dependency on a latent variable $x_{1:T}$ (as shown in the model in Figure 3.1). Basically, *EM* is an iterative algorithm, consisting of two steps:

1. *Expectation step.* The values of the unobserved latent variables are defined by calculating the probability of the latent variables, given the observed variables and the current values of the parameters. We compute the expected sufficient statistics.

2. *Maximization step.* The parameters are adjusted based on the defined latent variables.

Algorithm

The probability model is $p(y_{1:T}, x_{1:T}|\theta)$, where $y_{1:T}$ are observable and $x_{1:T}$ are not; θ represents the model parameters $\{A, B, C, D, F, G, \mathcal{Q}, \mathcal{R}, \mathcal{N}(\mu_0, \sum_0)\}$. $p(y_{1:T}|\theta)$ represents the marginal probability, so the log likelihood is:

$$
\begin{aligned}
l(\theta; x_{1:T}) &= log\, p(x_{1:T}|\theta) && (3.17)\\
&= log \sum_{y_{1:T}} p(y_{1:T}, x_{1:T}|\theta) && (3.18)\\
&= log \sum_{y_{1:T}} q(x_{1:T}|y_{1:T}) \frac{p(y_{1:T}, x_{1:T}|\theta)}{q(x_{1:T}|y_{1:T})} && (3.19)\\
&\geq \sum_{y_{1:T}} q(x_{1:T}|y_{1:T})\, log\, \frac{p(y_{1:T}, x_{1:T}|\theta)}{q(x_{1:T}|y_{1:T})} && (3.20)\\
&\triangleq \mathcal{L}\big(q(x_{1:T}), \theta\big) && (3.21)
\end{aligned}
$$

where the summation in equation (3.18) denotes marginalization, and $q(x_{1:T}|y_{1:T})$ is an averaging distribution. Equation (3.20) involves Jensen's inequality, due to the concavity of the logarithm function. $\mathcal{L}\big(q(x_{1:T}), \theta\big)$ is an auxiliary function that represents a lower bound for the log likelihood of an arbitrary distribution $q(x_{1:T}|y_{1:T})$. The *EM* algorithm is a coordinate ascent algorithm on the function $\mathcal{L}\big(q(x_{1:T}), \theta\big)$. At iteration $(t+1)$, we first maximize $\mathcal{L}\big(q(x_{1:T}), \theta^{(t)}\big)$ with respect to $q(x_{1:T}|y_{1:T})$. For this optimizing choice of averaging distribution $q^{(t+1)}$, we then maximize $\mathcal{L}\big(q(x_{1:T})^{(t+1)}, \theta^{(t)}\big)$ with respect to θ which yields the updated value $\theta^{(t+1)}$. Giving these steps their traditional names, we have:

$$
\begin{aligned}
q^{(t+1)} &= \mathbf{arg\,max}_{q(x_{1:T})}\, \mathcal{L}\big(q(x_{1:T}), \theta^{(t)}\big) && (3.22)\\
\theta^{(t+1)} &= \mathbf{arg\,max}_{\theta}\, \mathcal{L}\big(q(x_{1:T})^{(t+1)}, \theta\big) && (3.23)
\end{aligned}
$$

where equation (3.22) represents the *E* step and equation (3.23) the *M* step. (Appendix B Section B.1.2 shows how the optimization proceeds.)

[Ghahramani and Hinton, 1996] gives the expectation-maximization algorithm for estimating the parameters of linear systems represented by a state space model, equations (3.4)-(3.5). Rather than regarding the state as a deterministic value corrupted by random noise, the state and state noise variables are combined into a single Gaussian random variable. A similar idea was used for the observation variable. Based on equations (3.4) and (3.5) we can write the conditional densities for the state and output, $p(y_t|x_t)$ and $p(x_t|x_{t-1})$. Using equation (3.1) we can calculated the following equation (3.24) for the joint log probability, and generate the *EM* algorithm.

$$\begin{aligned}\log p(x_{0:T}, y_{1:T}) &= -\sum_{t=1}^{T}\left(\frac{1}{2}(y_t - Cx_t - Gu_t)'\mathcal{R}^{-1}(y_t - Cx_t - Gu_t)\right) - \frac{T\log|\mathcal{R}|}{2} \\ &\quad -\sum_{t=2}^{T}\left(\frac{1}{2}(x_t - Ax_{t-1} - Fu_t)'\mathcal{Q}^{-1}(x_t - Ax_{t-1} - Fu_t)\right) - \frac{(T-1)\log|\mathcal{Q}|}{2} \\ &\quad -\frac{1}{2}(x_0 - \mu_0)'\Sigma_0^{-1}(x_0 - \mu_0) - \frac{\log|\Sigma_0|}{2} - \frac{T(n_x + n_y)\log(2\pi)}{2} \end{aligned} \tag{3.24}$$

According to [Ghahramani and Hinton, 1996], the expectation-maximization algorithm can be though of as a forward-backward algorithm, where the forward part is computed by the Kalman filter and the backward part is computed using Raunch's recursion [Rauch *et al.*, 1965]. See [Ghahramani and Hinton, 1996] for the detailed procedure and formulas.

3.3.4 Least Squares Estimation

Control engineering approach

Modern control theory, starting around 1960, is an approach to the analysis and design of complex control systems. It is based on the concept of state and is applicable to multiple-input, multiple-output systems which may be linear or nonlinear, time invariant or time varying. Modern control theory is essentially a time-domain approach. Some important definitions in this context, using an engineering point of view, are state, state variables, state vector and state space, see Appendix B section B.1.3.

There are many techniques available for obtaining deterministic state space representations of discrete-time systems, [Ogata, 1995]. Consider the following very simple discrete-time system described by

$$y_t + a_1 y_{t-1} + \cdots + a_{n_a} y_{t-n_a} = b_0 u_t + b_1 u_{t-1} + \cdots + b_{n_b} u_{t-n_b} \tag{3.25}$$

where u_t is the input and y_t is the output of the system, as previously defined in section 3.2.1. The coefficients $\{a_i\}_{i=1}^{n_a}$ and $\{b_j\}_{j=1}^{n_b}$ are considered constant. This equation can also be written in the form of the pulse transfer function $G_p(z)$:

$$G_p(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_{n_b} z^{-n_b}}{1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}} \tag{3.26}$$

where $Y(z)$ represents the z-transform of y_t, $U(z)$ represents the z-transform of u_t, and $z = e^{sT_s}$ where s is the Laplace complex variable and T_s is sampling time. Using the direct programming method, equation (3.26) becomes

the following state-space representation, called a controllable canonical form, [Ogata, 1995].

$$\begin{bmatrix} x_t^1 \\ x_t^2 \\ x_t^3 \\ \cdot \\ \cdot \\ x_t^n \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n_a-1} & -a_{n_a} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{t-1}^1 \\ x_{t-1}^2 \\ x_{t-1}^3 \\ \cdot \\ \cdot \\ x_{t-1}^{n_a} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix} u_t \tag{3.27}$$

$$y_t = \left[\begin{array}{c|c|c|c} b_1 - a_1 b_0 & b_2 - a_2 b_0 & \cdots & b_{n_b} - a_{n_a} b_0 \end{array} \right] \begin{bmatrix} x_{t-1}^1 \\ x_{t-1}^2 \\ \cdot \\ \cdot \\ x_{t-1}^{n_a} \end{bmatrix} + [b_0] u_t \tag{3.28}$$

Equation (3.27) is a special case of equation (3.4). Here $\{x_t^i\}_{i=1}^{n_a}$ represents n_a states at time t, and the noise matrix is null ($B = 0$). Similarly, equation (3.28) is a special case of equation (3.5), where again the noise matrix is null ($D = 0$).

This state space representation, equations (3.27)-(3.28), is well-known in the control engineering community as a *deterministic* state space representation, where the states and observations are values corrupted by random noise.

Learning procedure

Based on the above approach, we can now use the conventional and well-known identification process for dynamic systems in the control engineering community.

Equation (3.25), commonly called the ARX model (Auto-Regressive with eXogenous variable) can be written as

$$\begin{aligned} y_t &= -a_1 y_{t-1} - \cdots - a_{n_a} y_{t-n_a} + b_0 u_t + b_1 u_{t-1} + \cdots + b_{n_b} u_{t-n_b} \\ &= \begin{bmatrix} -a_1 & -\cdots & -a_{n_a} & b_0 & \cdots & b_{n_b} \end{bmatrix} \begin{bmatrix} y_{t-1} \\ \cdots \\ y_{t-n_a} \\ u_t \\ \cdots \\ u_{t-n_b} \end{bmatrix} \\ &= f(y_{t-1}, \ldots, y_{t-n_a}, u_t, \ldots, u_{t-n_b}, \theta) \end{aligned} \tag{3.29}$$

Typically we have a set of training data $\{y_i, u_i\}_{i=1}^{n_{data}}$ from which to estimate the parameters $\theta = \{-a_1 \cdots - a_{n_a} b_0 \ldots b_{n_b}\}$. We use the *Least Squares Estimation* strategy, in which we pick θ in order to minimize the following Residual Sum of Squares (RSS):

$$RSS(\theta) = \sum_{i=min\{n_a,n_b\}}^{n_{data}} \left(y_i - f(y_{i-1}, \ldots, y_{i-n_a}, u_i, \ldots, u_{i-n_b}, \theta)\right)^2 \tag{3.30}$$

From a statistical point of view, this criterion is reasonable if the training observations $\{y_i, u_i\}_{i=1}^{n_{data}}$ represent independent random draws from their population. Even if they were not drawn randomly, the criterion is still valid if the observations are conditionally independent given the inputs.

If we rewrite the ARX model as $\mathcal{Y} = \mathcal{X}\Theta$ (see Appendix B Section B.1.3 for a detailed example), $RSS(\theta)$ can be rewritten as

$$RSS(\Theta) = (Y - \mathcal{X}\Theta)'(Y - \mathcal{X}\Theta) \tag{3.31}$$

This is a quadratic function in the number of parameters. Differentiating with respect to Θ we obtain

$$\frac{\partial RSS(\Theta)}{\partial \Theta} = -2\mathcal{X}'(Y - \mathcal{X}\Theta) \tag{3.32}$$

Assuming that $\mathcal{X}$ is nonsingular and hence $\mathcal{X}'\mathcal{X}$ is positive definite, we set the first derivative to zero

$$\mathcal{X}'(Y - \mathcal{X}\Theta) = 0 \tag{3.33}$$

to obtain the unique solution

$$\widehat{\Theta}_{LS} = (\mathcal{X}'\mathcal{X})^{-1}\mathcal{X}'Y \tag{3.34}$$

where $\widehat{\Theta}_{LS}$ is the best (*Least Squares*) estimator (see some properties of this estimator in Appendix B section B.1.3). A famous and important result in statistics asserts that $\widehat{\Theta}_{LS}$ has the smallest variance among all linear unbiased estimates, [Ljung, 1987]. With $\widehat{\Theta}_{LS}$ we can get the deterministic state space representation in the framework established in section 3.2.1.

3.4 Particle Filtering

If we want to work with probabilistic dynamic models which combine discrete and continuous states[4], such as the *JMLG* model (section 3.2.4), we have to work with algorithms for approximate inference. The state-of-the-art method is the Monte Carlo Particle Filter (*PF*). We can define the diagnosis/estimation task in a very simple way:

- *Given*: the observations (inputs and outputs) $\{u_t, y_t\}_{t=1}^{T}$, the *JMLG* model and its parameters,
- *Compute*: the most probable discrete mode $\{z_t\}_{t=1}^{T}$.

Given these observations, the inference task for any subset or property of the discrete modes $z_{0:t}$ relies on the joint probability distribution $p(z_{0:t}|y_{1:t}, u_{1:t})$. The original problem becomes that of *estimating* the distribution $p(z_{0:t}|y_{1:t}, u_{1:t})$[5] or some of its characteristics such as the filtering density $p(z_t|y_{1:t})$. The goal of the analysis is to compute the marginal posterior distribution of the discrete modes $p(z_{0:t}|y_{1:t})$. This distribution can be derived from the posterior $p(x_{0:t}, z_{0:t}|y_{1:t})$ by standard marginalization. The posterior density satisfies the following recursion:

$$p(x_{0:t}, z_{0:t}|y_{1:t}) = p(x_{0:t-1}, z_{0:t-1}|y_{1:t-1}) \times \frac{p(y_t|x_t, z_t)\, p(x_t, z_t|x_{t-1}, z_{t-1})}{p(y_t|y_{1:t-1})}. \tag{3.35}$$

However, this recursion involves intractable integrals. We must use numerical approximation schemes such as Particle Filtering. Particle Filtering computes, over time, a stochastic point-mass approximation of the posterior distribution of the states given the observations. (Some *PF* fundamentals are shown in Appendix B section B.2.1.)

[4] Generally assumed to be Gaussian distributed.

[5] For clarity, we drop off the control signal u_t from the argument of various probability distributions.

3.4.1 Algorithm

In the Particle Filtering setting, we use a weighted set of samples, called *particles*, $\{(x_{0:t}^{(i)}, z_{0:t}^{(i)}), w_t^{(i)}\}_{i=1}^N$ to approximate the posterior with the following point-mass distribution:

$$\widehat{p}_N(x_{0:t}, z_{0:t}|y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{x_{0:t}^{(i)}, z_{0:t}^{(i)}}(x_{0:t}, z_{0:t}), \tag{3.36}$$

where $\delta_{x_{0:t}^{(i)}, z_{0:t}^{(i)}}(x_{0:t}, z_{0:t})$ denotes the Dirac-delta function.

Given N particles $\{x_{0:t-1}^{(i)}, z_{0:t-1}^{(i)}\}_{i=1}^N$ at time $t-1$, approximately distributed according to the distribution $p(x_{0:t-1}^{(i)}, z_{0:t-1}^{(i)}|y_{1:t-1})$, *PF* enables us to compute N particles $\{x_{0:t}^{(i)}, z_{0:t}^{(i)}\}_{i=1}^N$ approximately distributed according to $p(x_{0:t}^{(i)}, z_{0:t}^{(i)}|y_{1:t})$ at time t. Since we cannot sample from the posterior directly, the Particle filtering update is accomplished by introducing an appropriate importance proposal distribution $q(x_{0:t}, z_{0:t})$ from which we can obtain samples. The pseudo-code for this basic algorithm is shown in Figure 3.10. The algorithm consists of two basic steps. We describe each one in detail.

- Sequential Importance Sampling (*SIS*)
- Selection

Sequential Importance Sampling (SIS) step

- For $i = 1, ..., N$, sample from the transition priors

$$\begin{aligned} \widehat{z}_t^{(i)} &\sim p(z_t|z_{t-1}^{(i)}) \\ \widehat{x}_t^{(i)} &\sim p(x_t|x_{t-1}^{(i)}, z_t^{(i)}) \end{aligned}$$

  and set $\left(\widehat{x}_{0:t}^{(i)}, \widehat{z}_{0:t}^{(i)}\right) \triangleq \left(\widehat{x}_t^{(i)}, \widehat{z}_t^{(i)}, x_{0:t-1}^{(i)}, z_{0:t-1}^{(i)}\right)$.

- For $i = 1, ..., N$, evaluate and normalize the importance weights

$$w_t^{(i)} \propto p\left(y_t|\widehat{x}_t^{(i)}, \widehat{z}_t^{(i)}\right)$$

Selection (Resampling) step

- Multiply/Discard particles $\left\{\widehat{x}_{0:t}^{(i)}, \widehat{z}_{0:t}^{(i)}\right\}_{i=1}^N$ with respect to high/low importance weights $w_t^{(i)}$ to obtain N particles $\left\{x_{0:t}^{(i)}, z_{0:t}^{(i)}\right\}_{i=1}^N$.

Figure 3.10: Standard sequential Monte Carlo algorithm at time t.

Sequential Importance Sampling

We have to extend the current paths $\{x_{0:t-1}^{(i)}, z_{0:t-1}^{(i)}\}_{i=1}^{N}$ to generate new paths at time t, $\{x_{0:t}^{(i)}, z_{0:t}^{(i)}\}_{i=1}^{N}$. We use the proposal distribution $q\left(\widehat{x}_{0:t}, \widehat{z}_{0:t}|y_{1:t}\right)$ given by the integral

$$q(\widehat{x}_{0:t}, \widehat{z}_{0:t}|y_{1:t}) = \int q(\widehat{x}_{0:t}, \widehat{z}_{0:t}|x_{0:t-1}, z_{0:t-1}, y_{1:t}) dp(x_{0:t-1}, z_{0:t-1}|y_{1:t-1}) \tag{3.37}$$

We propose

$$q(\widehat{x}_{0:t}, \widehat{z}_{0:t}|x_{0:t-1}, z_{0:t-1}, y_{1:t}) = q(\widehat{x}_t, \widehat{z}_t|x_{0:t-1}, z_{0:t-1}, y_{1:t}) \delta_{x_{0:t-1}, z_{0:t-1}}(\widehat{x}_{0:t-1}, \widehat{z}_{0:t-1}) \tag{3.38}$$

in order to leave the past trajectories intact. Only the current particles (at time t) are modified. We must do this to avoid the intractable integral in Equation (3.37). The resulting proposal distribution is:

$$q(\widehat{x}_{0:t}, \widehat{z}_{0:t}|y_{1:t}) = p(x_{0:t-1}, z_{0:t-1}|y_{1:t-1}) q(\widehat{x}_t, \widehat{z}_t|x_{0:t-1}, z_{0:t-1}, y_{1:t}) \tag{3.39}$$

Because we are sampling from $q(\widehat{x}_{0:t}, \widehat{z}_{0:t}|y_{1:t})$, the particles must be weighted by the importance weights

$$\begin{aligned} w_t &= \frac{p(\widehat{x}_{0:t}, \widehat{z}_{0:t}|y_{1:t})}{q(\widehat{x}_{0:t}, \widehat{z}_{0:t}|y_{1:t})} \\ &= \frac{p(dx_{0:t-1}, z_{0:t-1}|y_{1:t})}{p(dx_{0:t-1}, z_{0:t-1}|y_{1:t-1})} \times \frac{p(\widehat{x}_t, \widehat{z}_t|x_{0:t-1}, z_{0:t-1}, y_{1:t})}{q(\widehat{x}_t, \widehat{z}_t|x_{0:t-1}, z_{0:t-1}, y_{1:t})} \qquad (3.40) \\ &\propto \frac{p\left(y_t|\widehat{x}_t, \widehat{z}_t\right) p\left(\widehat{x}_t, \widehat{z}_t|x_{0:t-1}, z_{0:t-1}, y_{1:t}\right)}{q_t\left(\widehat{x}_t, \widehat{z}_t|x_{0:t-1}, z_{0:t-1}, y_{1:t}\right)}. \qquad (3.41) \end{aligned}$$

We can see that the optimal importance distribution, according to equation (3.40) is

$$q(\widehat{x}_t, \widehat{z}_t|x_{0:t-1}, z_{0:t-1}, y_{1:t}) = p(\widehat{x}_t, \widehat{z}_t|x_{0:t-1}, z_{0:t-1}, y_{1:t}). \tag{3.42}$$

However, equation (3.42) can be difficult to evaluate. We prefer to use the transition prior (which simplifies to the Markov density) as proposal distribution:

$$q(\widehat{x}_t, \widehat{z}_t|x_{0:t-1}, z_{0:t-1}, y_{1:t}) = p\left(\widehat{x}_t|x_{t-1}, z_{t-1}\right) p\left(\widehat{z}_t|z_{t-1}\right), \tag{3.43}$$

Equation (3.41) simplifies to the likelihood function

$$w_t \propto p\left(y_t|\widehat{x}_t, \widehat{z}_t\right). \tag{3.44}$$

Selection Step

The earliest Particle filtering implementations were based only on sequential importance sampling, which degenerates with time. [Gordon *et al.*, 1993] proposed a selection (or resampling) step which led to successful implementations.

The selection step eliminates samples with low importance weights and multiplies samples with high importance weights. A uniformly weighted posterior can be generated by resampling a set of uniformly weighted particles from the distribution represented by the weighted samples.

A selection or resampling scheme associates to each particle $(\widehat{x}_{0:t}^{(i)}, \widehat{z}_{0:t}^{(i)})$ a number of "children", say $N_i \in \mathbb{N}$, such that $\sum_{i=1}^{N} N_i = N$. There are various selection schemes such as

- Sampling importance re-sampling (*SIR*),
- Minimum variance sampling,
- Residual resampling.

All of them satisfy $\mathbb{E}(N_i) = N\widetilde{w}_t^{(i)}$, but their performance varies in terms of $var\,(N_i)$. Results in [Kitagawa, 1996; Crisan *et al.*, 1999] indicate that the restriction $\mathbb{E}(N_i) = N\widetilde{w}_t^{(i)}$ is unnecessary to obtain convergence. Considering these results, it is possible to design biased but computationally inexpensive selection schemes.

It was found that the specific choice of resampling scheme does not significantly after the performance of the particle filter[6]. We chose a minimum variance sampling algorithm. SIR and residual re-sampling are presented in Appendix B, section B.2.2.

Minimum variance sampling

This selection scheme includes stratified/systematic sampling procedures [Kitagawa, 1996] and the Tree Base Branching algorithm [Crisan, 2001]. A set of N points $\mathcal{U}$ are sampled in the interval $[0, 1]$, each of the points a distance $\frac{1}{N}$ apart. The number of children N_i is taken to be the number of points that lie between $\sum_{j=1}^{i-1} w_t^{(j)}$ and $\sum_{j=1}^{i} w_t^{(j)}$. This strategy introduces a variance on N_i, namely $var(N_i) = \overline{N}_t w_t'^{(i)}\left(1 - \overline{N}_t w_t'^{(i)}\right)$, where $w_t'^{(i)} = \overline{N}_t^{-1}\left(\widetilde{w}_t^{(i)} N - \widetilde{N}_i\right)$, $\widetilde{N}_i = \left\lfloor N\widetilde{w}_t^{(i)} \right\rfloor$ and $\overline{N}_t = N - \sum_{i=1}^{N} \widetilde{N}_i$. Its computational complexity is $\mathcal{O}(N)$.

A graphical representation of the Particle Filtering algorithm, using 10 particles, is given in Figure 3.11. In this example we show the following four sequential operations:

1. We start at time $t-1$ with a set of unweighted samples $\{\widehat{z}_{t-1}^{(i)}, \frac{1}{N}\}_{i=1}^{N}$ [7]. This set of samples provides an approximation of $p(z_{t-1}|y_{1:t-2})$.
2. For each sample, we can calculate its normalized importance weight using the information at time $t-1$, obtaining a set of weighted samples $\{\widehat{z}_{t-1}^{(i)}, \widetilde{w}_{t-1}^{(i)}\}_{i=1}^{N}$. This new set provides an approximation of $p(z_{t-1}|y_{1:t-1})$.
3. The selection step selects the most important particles to obtain a set of unweighted particles $\{\widehat{z}_{t-1}^{(i)}, \frac{1}{N}\}_{i=1}^{N}$. This new set is still an approximation of $p(z_{t-1}|y_{1:t-1})$.
4. The sampling step generates variety. We obtain a new set of particles $\{\widehat{z}_t^{(i)}, \frac{1}{N}\}_{i=1}^{N}$ at time t which approximates $p(z_t|y_{1:t-1})$.

3.5 Rao-Blackwellized Particle Filtering

Rao-Blackwellized Particle Filtering (*RBPF*) [Akashi and Kumamoto, 1977; Doucet *et al.*, 2000a; de Freitas, 2001] is a Particle Filtering variant which exploits some of the analytical structure of the model. Basically, if we know the values of the discrete modes z_t, it is possible to compute the distribution of the continuous states x_t exactly. We can therefore combine a Particle filter to compute the distribution of the discrete modes with a bank of Kalman filters to compute the distribution of the continuous states. That is, we approximate the posterior distribution with a recursive, stochastic mixture of Gaussians. This strategy is known as *Rao-Blackwellization* because it is related to the *Rao-Blackwell* formula [Casella and Robert, 1996], (Appendix B section B.3.1).

[6] We tried the three selection algorithms with similar results.

[7] For clarity, the graph is only showing 1 variable $\{\widehat{z}_{t-1}^{(i)}\}$, but we are working with both $\{\widehat{x}_{t-1}^{(i)}, \widehat{z}_{t-1}^{(i)}\}$.

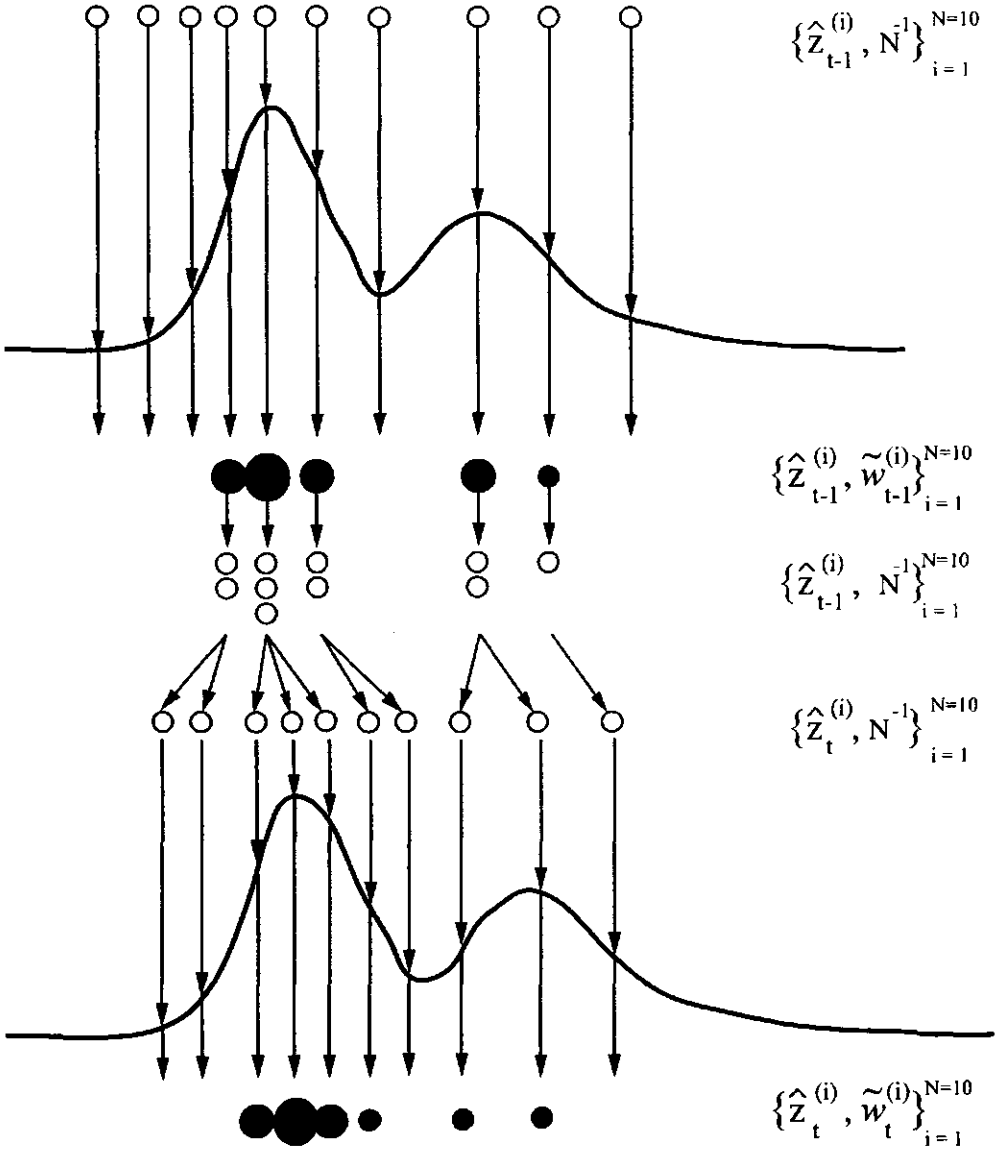

Figure 3.11: Standard Particle Filter. An approximation of $p(z_{t-1}|y_{1:t-2})$ is obtained through unweighted measure $\{\hat{z}_{t-1}^{(i)}, \frac{1}{N}\}_{i=1}^{N}$ at time $t-1$. For each particle the importance weights are computed at time $t-1$, $\{\hat{z}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\}_{i=1}^{N}$, which generates an approximation of $p(z_{t-1}|y_{1:t-1})$. The selection step is applied and an approximation of $p(z_{t-1}|y_{1:t-1})$ is obtained using unweighted particles $\{\hat{z}_{t-1}^{(i)}, \frac{1}{N}\}_{i=1}^{N}$. Note that this approximated distribution and the previous one are the same. The *SIS* step yields $\{\hat{z}_{t}^{(i)}, \frac{1}{N}\}_{i=1}^{N}$ which is an approximation of $p(z_t|y_{1:t-1})$ at time t.

Following the *Rao-Blackwell* formula, we can factorize

$$p(x_{0:t}, z_{0:t}|y_{1:t}) = p(x_{0:t}|y_{1:t}, z_{0:t})\, p(z_{0:t}|y_{1:t}), \tag{3.45}$$

The density $p(x_{0:t}|y_{1:t}, z_{0:t})$ in equation (3.45) is Gaussian and can be computed analytically if we know the marginal posterior density $p(z_{0:t}|y_{1:t})$. We can compute this density recursively using

$$p(z_{0:t}|y_{1:t}) = p(z_{0:t-1}|y_{1:t-1}) \times \frac{p(y_t|y_{1:t-1}, z_{0:t})\, p(z_t|z_{t-1})}{p(y_t|y_{1:t-1})} \tag{3.46}$$

Equation (3.46) does not have a closed-form solution. If we attempt to solve this problem analytically, we obtain intractable integrals, hence a numerical approximation must be used. It is important to point out that the density $p(y_t|y_{1:t-1}, z_{0:t})$ in equation (3.46) does not simplify to $p(y_t|z_t)$, as does its equivalent in equation (3.35), because there is a dependency on past values through $x_{0:t}$.

Using a set of weighted samples $\{z_{0:t}^{(i)}, w_t^{(i)}\}_{i=1}^N$ to represent the marginal posterior distribution, we can approximate equation (3.46) using

$$\widehat{p}_N(z_{0:t}|y_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta_{z_{0:t}^{(i)}}(z_{0:t}), \tag{3.47}$$

The marginal density of $x_{0:t}$ is a Gaussian mixture

$$\begin{aligned}
\widehat{p}_N(x_{0:t}|y_{1:t}) &= \int p(x_{0:t}|z_{0:t}, y_{1:t}) dp(z_{0:t}|y_{1:t}) \\
&= \int p(x_{0:t}|z_{0:t}, y_{1:t}) \sum_{i=1}^{N} w_t^{(i)} \delta_{z_{0:t}^{(i)}}(z_{0:t}) \\
&= \sum_{i=1}^{N} w_t^{(i)} p(x_{0:t}|y_{1:t}, z_{0:t}^{(i)})
\end{aligned}$$

This Gaussian mixture can be computed efficiently with a stochastic bank of Kalman filters.

3.5.1 Algorithm

The *Rao-Blackwellized Particle Filtering* algorithm, whose pseudo-code appears in Figure 3.12, consists of three basic steps:

1. sequential importance sampling,
2. selection step, and
3. updating step.

We briefly discuss each step.

Sequential Importance Sampling

We sample $z_t^{(i)}$ and then propagate the mean $\mu_t^{(i)}$ and covariance $\Sigma_t^{(i)}$ of x_t with a Kalman filter. Table 3.1 shows the Kalman filter formulation. Since the dimension of $p(z_{0:t}|y_{1:t})$ is smaller than that of $p(z_{0:t}, x_{0:t}|y_{1:t})$, used in standard Particle Filtering, we should expect to obtain better results with less variance. See Appendix B section B.3.2.

Kalman Filter.

The Kalman Filter (*KF*) is a set of mathematical equations, Table 3.1, that implement a *predictor-corrector* type estimator which is *optimal* in the sense that it minimizes the estimated error covariance. It is one of the most well-known and often-used tools for stochastic estimation using noisy sensor measurements.

The Kalman filter estimates a process by using a form of feedback control. The filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups:

- *time update* equations, and
- *measurement update* equations.

The time update equations are responsible for projecting the current state (equation 1 in Table 3.1) and error covariance (equation 2) estimates forward in time to obtain the a priori estimates for the next time step. The measurement update equations (5 and 6) are responsible for the feedback - i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

The time update equations can also be thought of as predictor equations, while the measurement update equations can be though of as corrector equations. Indeed the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems.

If the process to be estimated is non-linear, or the measurement relationship to the process is non-linear, or both, then we have to use the *extended Kalman Filter (EKF). EKF* is a Kalman filter which linearizes, using the partial derivatives of the process and measurement functions, about the current mean and covariance.

Table 3.1: Kalman Filter equations.

Equation	Formula	Definition
1	$\mu_{t\|t-1}^{(i)} = A(z_t^{(i)})\mu_{t-1\|t-1}^{(i)} + F(z_t^{(i)})u_t$	$\mu_{t\|t-1} \triangleq \mathbb{E}(x_t\|y_{1:t-1})$
2	$\Sigma_{t\|t-1}^{(i)} = A(z_t^{(i)})\Sigma_{t-1\|t-1}^{(i)}A(z_t^{(i)})^{\mathsf{T}} + B(z_t^{(i)})B(z_t^{(i)})^{\mathsf{T}}$	$\Sigma_{t\|t-1} \triangleq cov(x_t\|y_{1:t-1})$
3	$S_t^{(i)} = C(z_t^{(i)})\Sigma_{t\|t-1}^{(i)}C(z_t^{(i)})^{\mathsf{T}} + D(z_t^{(i)})D(z_t^{(i)})^{\mathsf{T}}$	$S_t \triangleq cov(y_t\|y_{1:t-1})$
4	$y_{t\|t-1}^{(i)} = C(z_t^{(i)})\mu_{t\|t-1}^{(i)} + G(z_t^{(i)})u_t$	$y_{t\|t-1} \triangleq \mathbb{E}(y_t\|y_{1:t-1})$
5	$\mu_{t\|t}^{(i)} = \mu_{t\|t-1}^{(i)} + \Sigma_{t\|t-1}^{(i)}C(z_t^{(i)})^{\mathsf{T}}S_t^{-1(i)}(y_t - y_{t\|t-1}^{(i)})$	$\mu_{t\|t} \triangleq \mathbb{E}(x_t\|y_{1:t})$
6	$\Sigma_{t\|t}^{(i)} = \Sigma_{t\|t-1}^{(i)} - \Sigma_{t\|t-1}^{(i)}C(z_t^{(i)})^{\mathsf{T}}S_t^{-1(i)}C(z_t^{(i)})\Sigma_{t\|t-1}^{(i)}$	$\Sigma_{t\|t} \triangleq cov(x_t\|y_{1:t})$

Importance distribution. Using the prior proposal for z_t and applying equation (3.46), we find that the importance weights for z_t are given by the predictive density

$$p(y_t|y_{1:t-1}, z_{0:t}) = \mathcal{N}(y_t; y_{t|t-1}, S_t). \tag{3.48}$$

This is the most widely used distribution (as with the standard Particle Filter) because it is easy to compute. However, it can be inefficient since it ignores the most recent evidence.

Selection step

As in the standard Particle Filter, we use the resampling procedure in order to avoid the degeneracy of the sequential importance sampling simulation method.

Updating step

We perform one step of the Kalman recursion to compute the minimum statistics $\left\{\mu_{t+1|t}^{(i)}, \Sigma_{t+1|t}^{(i)}, y_{t+1|t}^{(i)}, S_{t+1}^{(i)},\right\}$ given $\left\{z_t^{(i)}, \mu_{t|t-1}^{(i)}, \Sigma_{t|t-1}^{(i)}\right\}$. Basically, we use equations 5 and 6 in Table 3.1 with the fittest particles after the resampling step.

The Rao-Blackwellized *PF* estimate is usually computationally more expensive than the standard Particle Filter. Intuitively, we expect it is worth performing Rao-Blackwellized Particle Filtering when the average conditional variance of the variable $x_{0:t}$ is high.

Sequential importance sampling step

- For $i = 1, ..., N$, set $\widehat{\mu}_{t|t-1}^{(i)} \triangleq \mu_{t|t-1}^{(i)}, \widehat{\Sigma}_{t|t-1}^{(i)} \triangleq \Sigma_{t|t-1}^{(i)}$, sample

$$\widehat{z}_t^{(i)} \sim \Pr(z_t | z_{t-1}^{(i)})$$

  and set

$$\widehat{z}_{0:t}^{(i)} \triangleq \left(\widehat{z}_t^{(i)}, z_{0:t-1}^{(i)}\right)$$

- For $i = 1, ..., N$, evaluate and normalize the importance weights

$$w_t^{(i)} \propto p\left(y_t | y_{1:t-1}, \widehat{z}_t^{(i)}\right)$$

Selection step (Resampling)

- Multiply/Discard particles $\left\{\widehat{\mu}_{t|t-1}^{(i)}, \widehat{\Sigma}_{t|t-1}^{(i)}, \widehat{z}_t^{(i)}\right\}_{i=1}^{N}$ with respect to high/low importance weights $w_t^{(i)}$ to obtain N particles $\left\{\mu_{t|t-1}^{(i)}, \Sigma_{t|t-1}^{(i)}, z_t^{(i)}\right\}_{i=1}^{N}$.

Updating step

- For $i = 1, ..., N$, use one step of the Kalman recursion to compute the minimum statistics $\left\{\mu_{t+1|t}^{(i)}, \Sigma_{t+1|t}^{(i)}, y_{t+1|t}^{(i)}, S_{t+1}^{(i)},\right\}$ given $\left\{z_t^{(i)}, \mu_{t|t-1}^{(i)}, \Sigma_{t|t-1}^{(i)}\right\}$.

Figure 3.12: Rao-Blackwellized Particle Filtering algorithm at time t. RBPF combines Particle Filtering to compute the distribution of the discrete modes z_t with a bank of Kalman filters to compute the distribution of the continuous states x_t.

3.6 Summary

In this chapter we reviewed the fundamentals on which our research is based.

We reviewed two well-known graphical models, the *state space model* and the *hidden Markov model*. The *Jump Markov Linear Gaussian model* (*JMLG*) is a combination of these. We tested the *JMLG* model with real process data and found that it is an excellent framework for representing linear dynamic systems which exhibit different behaviours (discrete modes) over time. We show these results in Chapter 5. Unfortunately, there is no procedure for learning the full set of *JMLG* parameters. In the following Chapter 4 we propose an algorithm to cope with this problem. Our proposal is based on the *Expectation-Maximization* and *Least Squares Estimation* methods that we discussed in this chapter.

We presented the Particle Filtering (*PF*) setting, describing a Markov chain Monte Carlo algorithm. This algorithm approximates the belief state using a set of particles and keeps the distribution updated over time. We also discussed Rao-Blackwellized Particle Filtering (*RBPF*), a Particle Filtering variant, which exploits some of the analytical structure of the *JMLG* model. *RBPF* typically exhibits results with lower variance than standard *PF*. We gave the basic steps and principles of these algorithms (most of the proofs and details are included in Appendix B). Both *PF* and *RBPF* were implemented and tested with several real-world processes (Chapter 5 and 6). Based on the theoretical foundations discussed here and the experimental results of *PF* and *RBPF*, we propose a new Particle Filtering variant which can be implemented in real-time for fault diagnosis or state estimation in dynamic systems.

Chapter 4

Look-ahead Rao-Blackwellized Particle Filtering

4.1 Introduction

Complex systems with high demands on performance and availability are the result of modern technology. Fault-tolerant control, monitoring and diagnosis capabilities play a crucial role in achieving these requirements. Domains include autonomous systems (i.e. spacecraft, planetary rovers, and mobile robots), industrial plants (i.e. chemical plants, oil refineries, and nuclear plants), etc. Monitoring and diagnosis algorithms for these applications must be efficient and preferably on-line. We need quick diagnosis (usually real-time) using limited information and computational resources.

We can view the diagnosis task as an estimation problem in which we must determine the state of a system over time, given some data from that system. Model-based diagnosis is a common approach to this problem, in which the overall system state is represented as an assignment of a mode to each component of the system. This assignment is an alternative description of the state, if the set of models associated with the modes is consistent with the observations. However, model-based diagnosis traditionally operates on discrete modes only, hence continuous variables must be discretized. During the discretization process, important information is generally lost, and transient events cannot be diagnosed. Reasoning is inadequate with discretized models because fine discretization is required to accurately model the complex dynamics. Continuous behaviours are difficult to capture using the pure qualitative models of discrete model-based reasoning systems. Reasoning in terms of continuous dynamics is important if we want to detect functional failures, as well as low-level incipient faults and subtle component degradation. To overcome these problems, we need to reason directly with continuous variables.

To tackle this problem we must address some issues. A system's continuous dynamics and its evolution through different behaviour modes are coupled. For this reason, we need *hybrid* monitoring and diagnosis capability. We need to monitor a system's performance along both its discrete mode changes and its continuous state changes.

A *hybrid system* considers a *set of discrete modes*, where a mode represents a fault state or operational condition of the system, and a *set of continuous variables* which model the continuous quantities of the system. In this context, we use the term *hybrid state* to talk about a mode of the system and a value for each continuous variable of the system. With each observable variable we associate an observation function that defines the likelihood of an observation given the mode and the values of the continuous variables. The observation function is required

because generally not all the states are observable. Finally, we must formally represent the noises associated with both continuous and discrete variables. The jump Markov linear Gaussian model (Chapter 3 section 3.2.4) is used to model this hybrid system. Learning the *JMLG* model's parameters is one of the main problems to solve.

To detect the onset of subtle failures, it is essential that a monitoring and diagnosis system be able to accurately extract the hybrid state of the system from a signal that may be hidden among stochastic disturbances such as measurement noise. This is the diagnosis task or hybrid estimation problem we want to solve.

Computing exact hybrid diagnoses in this probabilistic dynamic model is computationally intractable in the general case. The most general approaches to overcoming this problem are the Particle filtering algorithms, which sequentially compute a numerical approximation to the posterior probability distribution of the system states given the observations.

To solve the hybrid diagnosis problem we are proposing *Look-ahead Rao-Blackwellized Particle Filtering* (*la-RBPF*), an efficient variant of the Particle Filtering algorithms. Basically, we are using particles to estimate the distribution of discrete modes, and we are approximating the continuous variables using a multivariate Gaussian that is updated at each time-step using a Kalman filter.

In order to produce an effective diagnosis algorithm, we have to solve several problems such as very low prior fault probabilities, nonlinear behaviour, and high dimensional state spaces.

Because Particle Filters are numerical approximation algorithms based on sampling, they have particular trouble with diagnosis problems due to the low probabilities of transitions to fault states. If a fault state has a very low prior, then a very small number of particles will enter the state. A classical heuristic solution to this problem is to increase the number of particles; however, this may increase the computing time beyond real-time requirements. Most industrial applications demand real-time diagnosis with limited computational resources. Our proposal is efficient with fewer particles.

la-RBPF is restricted to linear models with Gaussian noise. Most real systems have nonlinear behaviour; however, we can break nonlinear states into several pieces that are approximately linear Gaussian. Unfortunately, as the number of possible faults grows, or nonlinear states are broken into linear Gaussian models, the number of discrete modes in the system grows. A high number of modes requires a huge number of particles to accurately approximate the posterior distribution. The number of particles also grows exponentially with the dimensionality of the continuous states.

In this chapter, our hybrid model is formally described in section 4.2.1, where we establish the meaning of each parameter and its relationship with the diagnosis/estimation problem in this context. A learning algorithm based on the Least Squares Estimation algorithm and Expectation-Maximization method is proposed in section 4.2.3. Our main contribution, the inference task is described in section 4.3, and the *look-ahead Rao-Blackwellized Particle Filtering* (*la-RBPF*) inference algorithm is presented in section 4.3.3. The fundamentals of *la-RBPF* were presented in Chapter 3, so here the algorithm's description is straightforward. Finally, some expected results are summarized in Section 4.4. These results are demonstrated via experimental and simulated tests in Chapters 5-6, respectively.

4.2 Modelling

4.2.1 Model Formulation

Formally, we can represent the hybrid system to be diagnosed using the *jump Markov Linear Gaussian* model (*JMLG*), adopted by [de Freitas, 2001] (briefly described in Chapter 3, section 3.2.3)

$$\begin{aligned} z_t &\sim p(z_t|z_{t-1}) \\ x_{t+1} &= A(z_{t+1})x_t + B(z_{t+1})\gamma_{t+1} + F(z_{t+1})u_{t+1} \\ y_t &= C(z_t)x_t + D(z_t)\upsilon_t + G(z_t)u_t, \end{aligned}$$

where:

- $z_t \in \{1, \ldots, n_z\}$ denotes the unknown possible discrete modes the system can be in (normal operation, faulty conditions, etc.) n_z represents the number of possible discrete modes. The variable z_t denotes a discrete-time, time-homogeneous, first-order Markov chain at time t.
- $x_t \in \mathbb{R}^{n_x}$ denotes the unknown continuous states the system can be in. n_x represents the number of continuous states.
- $y_t \in \mathbb{R}^{n_y}$ denotes the continuous observable variables. n_y represents the number of measurements.
- $u_t \in \mathbb{R}^{n_u}$ is another continuous observable variable, a generally known control signal or exogenous input sent to the system at time t. n_u represents the number of inputs. Generally, $u_t \in \mathcal{U}$.
- $\gamma_t \in \mathbb{R}^{n_\gamma}$ is an exogenous input called the process noise. n_γ is the number of process noises (usually $n_\gamma = n_x$). γ_t can be modelled as a random, uncorrelated sequence with zero mean and Gaussian distribution specified by the covariance matrix. Mathematically, γ_t is *iid* Gaussian, $\gamma_t \sim \mathcal{N}(0, I_{n_{gamma}})$, with covariance $\mathbb{E}[\gamma_t \gamma_t'] = \mathcal{Q}$.
- $\upsilon_t \in \mathbb{R}^{n_\upsilon}$ is an exogenous input called the measurement noise. n_υ is the number of measurement noises (usually $n_\upsilon = n_y$). υ_t is *iid* Gaussian, $\upsilon_t \sim \mathcal{N}(0, I_{n_\upsilon})$ with covariance $\mathbb{E}[\upsilon_t \upsilon_t'] = \mathcal{R}$.
- $p(z_t|z_{t-1})$ is a probabilistic transition function over the discrete modes z_t. $p_{i,j} \triangleq Pr(z_t = j|z_{t-1} = i)$ for any $i, j \in \{1, \ldots, n_z\}$. The transition probability matrix is in $\mathbb{R}^{n_z \times n_z}$, with elements satisfying $p_{i,j} \geq 0$ and $\sum_{j=1}^{n_z} p_{i,j} = 1$ for each row $i \in \{1, \ldots, n_z\}$.
- The initial states are:
  - the prior distribution $z_0 \sim p(z_0)$ for the discrete modes
  - $x_0 \sim \mathcal{N}(\mu_0, \sum_0)$ for the continuous states, where $\sum_0 > 0$.
- The parameters $\big(A(z_t), B(z_t), C(z_t), D(z_t), F(z_t), G(z_t)\big)$[1] are matrices with $D(z_t)D(z_t)' > 0$ for any $z_t \in \{1, \ldots, n_z\}$.

We state explicitly that our model implies the following continuous densities:

$$\begin{aligned} p(x_t|z_t, x_{t-1}, u_t) &= \mathcal{N}\big(A(z_t)x_{t-1} + F(z_t)u_t, B(z_t)B(z_t)'\big) \\ p(y_t|z_t, x_t, u_t) &= \mathcal{N}\big(C(z_t)x_t + G(z_t)u_t, D(z_t)D(z_t)'\big) \end{aligned}$$

Figure 4.1 shows a general graphical representation of the proposed hybrid system, using the jump Markov linear Gaussian (*JMLG*) model[2], repeated here for convenience. G matrix is null for all our domains, so the simplified graphical representation for the jump Markov linear Gaussian (*JMLG*) model is shown in Figure 4.2.

[1] The matrix G is a null matrix; however, they are considered for completeness.

[2] For clarity, we omit the noise signals.

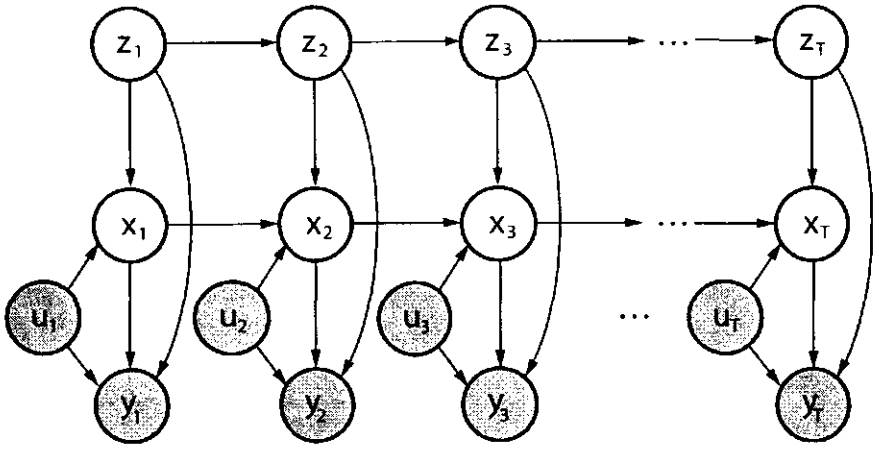

Figure 4.1: General JMLG model for hybrid system modelling. y_t is the measurement variable. u_t is the control signal. x_t is the continuous state. z_t is the discrete mode, which follows the Markovian property. Shadow nodes represent observable (known) variables. x_0 and z_0 (not shown here) represent the initial continuous states and discrete mode, respectively.

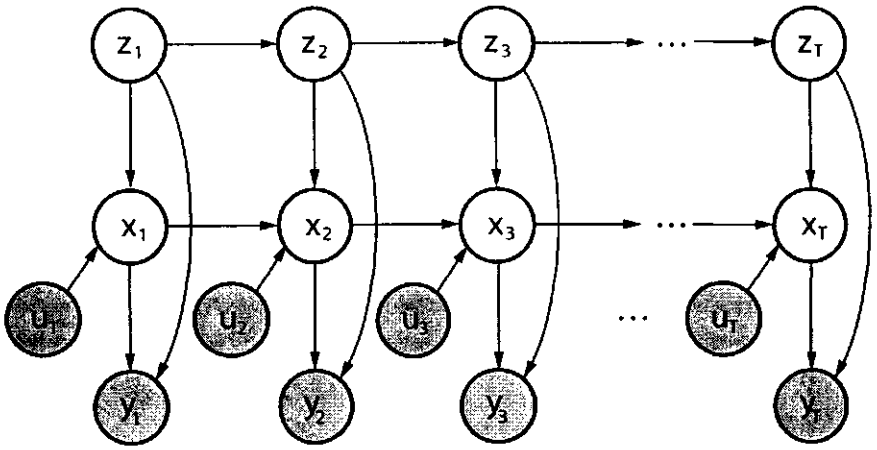

Figure 4.2: JMLG model for hybrid system modelling. There is not direct dependence between the measurement variable y_t and the control signal u_t.

4.2.2 Problem definition

The problem of learning the parameters of a deterministic state space model is known in the control engineering community as the system identification problem. In its most general form it assumes access only to sequences of input variables $u_{1:t}$ and output $y_{1:t}$ observations (shadow nodes shown in Figure 4.1). This problem has a well-known solution if we have only one discrete mode ($n_z = 1$). However, modelling the parameters for the general *JMLG* model represents an intractable problem. Table 4.1 shows the known and unknown variables. We can formulate the modelling problem as follows:

Given the data $u_{1:T}$ and $y_{1:T}$ and the dimensions n_u and n_y of this data, find the parameters $\theta = \big(n_z, n_x, \{A(j), B(j), C(j), D(j), F(j), G(j)\}_{j=1}^{n_z}, p(z_t|z_{t-1}), p(z_0), \mathcal{N}(\mu_0, \sum_0)\big)$ for the jump Markov linear Gaussian model.

Table 4.1: JMLG model formulation.

known variables	unknown variables
n_y	$\big(A(j), B(j), C(j), D(j), F(j), G(j)\big)_{j=1}^{n_z}$
$y_{1:T}$	n_z
n_u	n_x
$u_{1:T}$	$p(z_t\|z_{t-1}), p(z_0), \mathcal{N}(\mu_0, \sum_0)$

4.2.3 Learning algorithm

As we pointed out in Chapter 3, Section 3.3.2, this learning problem is unsolved. The solution is intractable for different reasons, and particularly because there is a huge number $n_z \leq T$ of possible discrete modes and a potentially infinite number n_x of continuous state variables. This leads to

- Huge possible sequences $\{z_t\}_{t=0}^{T}$, $z_t \in \{1, \ldots, n_z\}$.
- Huge possible transition matrices $p(z_t|z_{t-1}) \in \mathbb{R}^{n_z \times n_z}$ and $p(z_0) \in \mathbb{R}^{n_z \times 1}$.
- Huge number of possible matrices $\{A(j), B(j), C(j), D(j), F(j)\}_{j=1}^{n_z}$; indeed an infinite number for $\{A(j), B(j), F(j)\}_{j=1}^{n_z} \in \mathbb{R}^{n_x \times n_x}$.

In order to solve this problem, we are proposing a practical solution strongly based on knowledge about the process. Note that the end goal is not to find the best possible *JMLG* model. Our main goal is to build a good *JMLG* model of the process in order to estimate/diagnose its operating condition. Our proposal is as follows.

STEP 1 (Discrete Modes)

Based on our process knowledge and inference goals, we identify the number of possible discrete modes n_z. The number n_z of discrete modes must be representative of the most probable faulty points (i.e. stuck wheel, faulty sensor, broken valve), operating modes (i.e. normal condition, startup, shutdown) or nonlinear regimes that the process or system can be in. We have to define a discrete mode for each combination of faults, operating modes or nonlinear regimen. For simultaneous faults, the number of discrete modes grows exponentially. This number n_z defines the main dimensionality of the problem.

STEP 2 (Markov Properties)

We design the transition matrix $p(z_t|z_{t-1}) \in \mathcal{R}^{n_z \times n_z}$ based on historical data for the process/system. This transition matrix must be representative of how the operating conditions change over time or how the faults can appear. The design of this transition matrix depends on the problem to be solved. For the purposes of this research the transition matrices were engineered based on our experience of the process. We must also define the prior distribution $z_0 \sim p(z_0) \in \mathcal{R}^{n_z \times 1}$, which indicates the most probable initial condition for each discrete mode. The results of this step are:

- $p(z_0)$, initial prior distribution for the discrete modes
- $p(z_t|z_{t-1})$, transition matrix for the discrete modes

STEP 3 (Identification)

Identification (learning) is a method for model building based on experiments. Basically, it is the determination, on the basis of input and output, of the model within a specified class of models to which the system under test is equivalent. By this definition three basic entities are involved in system identification: the data, a set of models, and a rule or criterion for model estimation. The data are usually obtained from suitably designed experiments. Then a suitable model must be determined within the set of candidate models. Finally the model is tested to see whether it is an appropriate representation of the system. See Appendix C, section C.1.1 for the general procedure of system identification.

We have to learn the parameters (A, B, C, D, F, G) for each of the n_z discrete modes of the jump Markov linear Gaussian model. It is very important to note that given n_z, $p(z_t|p_{z-1})$ and $p(z_0)$, the new problem is relatively easy to tackle. We have divided the original learning problem into n_z small learning problems, each to be carried out using the following procedure, usually iterative.

Step 3a (Design the experimental tests)

We must design the experimental test for each discrete mode. We can divide this stage into two sequential steps:

1. Step-Response analysis is implemented in order to identify basic process information such as the range of linearity of the process, static gains and relevant time constants (dead time and delay time). Step responses can furnish this information to a sufficient degree of accuracy. Based on plots of the step-response, some characteristic numbers can be graphically constructed, which in turn can be used to determine parameters in a model of given order. See [Smith and Corripio, 1997] for a complete description.
2. We then use the information from the step-response analysis to design frequency-rich input signals such as the Pseudo Random Binary Sequence (*PRBS*). PRBS tests follow the persistent excitation condition. *PRBS* tests have been widely used in the control engineering community for system identification. See Appendix C, section C.1.3 for a detailed description of Pseudo Random Binary Sequence tests.

The results of this step, for each discrete mode, are:

- n_y, the dimension of the continuous measurement variables
- n_u, the dimension of the control signal

- $\{u_i\}_{i=1}^{n_{data}}$, the experimental control signal sequence
- $\{y_i\}_{i=1}^{n_{data}}$, the experimental measurements sequence

Step 3b (Learning the deterministic model)

Given the sequence $\left(\{u_i, y_i\}_{i=1}^{n_{data}}\right)$ for each discrete mode, we must now learn the parameters and dimension of the deterministic model. The simplest and most common model follows the Auto-Regressive with eXogenous variable structure (*ARX*). For the *ARX* structure we must define the following numbers of parameters. These numbers are closely related to the dimension n_x of the continuous states matrices:

1. n_a, the number of a coefficients (auto-regressive side)
2. n_b, the number of b coefficients (exogenous side)

After defining these numbers, the parameter estimation problem, $\{a_i\}_{i=1}^{n_a}$ and $\{b_j\}_{j=1}^{n_b}$, of linear parameter models can be computed. This can be treated as a well-known linear regression problem (see Chapter 3, section 3.3.4 for a complete description, or [Ljung, 1987] for other methods). The definition of the model structure and computation of parameters $\{a_i\}_{i=1}^{n_a}$ and $\{b_j\}_{j=1}^{n_b}$ is an iterative procedure. The procedure ends when some condition is reached, (see Chapter C, section C.1.4).

The ARX model can be translated into a deterministic state space representation (Appendix B, section B.1.3). The main results of this step are:

- n_x, the dimension of the continuous states
- The matrices $\{A(j) \in \mathbb{R}^{n_x \times n_x}, C(j) \in \mathbb{R}^{n_y \times n_x}, F(j) \in \mathbb{R}^{n_x \times n_u}, G(j) = 0 \in \mathbb{R}^{n_y \times n_u}\}_{j=1}^{n_x}$

Step 3c (Learning the probabilistic model)

In this second learning step, we use the Expectation-Maximization (*EM*) algorithm, (Chapter 3 section 3.3.3). This stage is needed to compute the noise matrices (B, D) and refine the estimates of matrices (A, C, F, G) computed in the previous stage.

The Expectation-Maximization algorithm consists of two steps:

- the *E* step: a *Rauch-Tung-Striebel* Kalman smoother is used to compute the sufficient statistics of the Gaussian states,
- the *M* step: we update the matrices of parameters using analytically derived equations.

The *EM* results strongly depend on the initial conditions. For this reason, we use the previous learning stage to get a very good initial approximation of the matrices A, C, F, G. This initial approximation contributes significantly toward avoiding convergence to shallow local maximum of the likelihood function. Note that we do not need additional experimental data in this step; the data generated by the PRBS test are persistent input signals, and the *EM* algorithm can find the right solution. The initialization step with the deterministic model also provides us with rich information about the dynamic behaviour of each discrete mode. It might be possible to replace this initialization step with another method, such as simulated annealing; however, we prefer a more elegant step exploiting the process knowledge.

The main results of this step are:

- The matrices $\{B(j) \in \mathbb{R}^{n_x \times n_\gamma}, D(j) \in \mathbb{R}^{n_y \times n_\upsilon}\}_{j=1}^{n_z}$
- Re-computed matrices $\{A(j) \in \mathbb{R}^{n_x \times n_x}, C(j) \in \mathbb{R}^{n_y \times n_x}, F(j) \in \mathbb{R}^{n_x \times n_u}, G(j) = 0 \in \mathbb{R}^{n_y \times n_u}\}_{j=1}^{n_z}$
- Initial states $x_0 \sim \mathcal{N}(\mu_0, \sum_0)$ for each of the n_z discrete modes.

Step 3d (Validation)

After steps 3a, 3b and 3c, there are standard validation procedures, but we have to keep in mind that the obtained parameters for the *JMLG* model represent a good estimation within the chosen model structure (n_x, n_y, n_u, etc.) The crucial question is whether the learned model is *good enough* for the inference problem. We have to verify that the learned model agrees with the real process behaviour; see Appendix C, section C.1.4 for some guidelines. If the learned model does not agree with the real process, we must (in the following order):

1. Try other initial conditions for the *EM* algorithm (probabilistic model)
2. Increase the model complexity for the Least Squares Estimate algorithm (deterministic model)
3. Try another experimental data set (representative data)

Figure 4.3 shows the pseudo-code of the algorithm for learning the parameters of the *JMLG* model. This proposal assumes the designer can take advantage of good process knowledge. Figure 4.4 presents a simple flow diagram of this procedure. Between each step we show the resulting information and the iteration loops.

Figure 4.5 shows a graphical example of the obtained results (omitting the meaning of each variable). The upper graph represents the discrete modes over time. The middle graph shows the real data and the deterministic state space representation.

4.3 Inference problem

4.3.1 Problem definition

Given a jump Markov linear Gaussian model, a sequence of observations $y_{1:t}$ and control inputs $u_{1:t}$, estimate (on-line) the most likely hybrid state $\{z_t, x_t\}$ at each time t. Essentially, we want to find the discrete modes (Figure 4.5, upper graph) given the observations (middle graph) over time.

As we established in Chapter 3 section 3.4, the inference task for any property of the discrete modes and continuous states relies on the joint probability distribution $p(x_{0:t}, z_{0:t}|y_{1:t}, u_{1:t})$[3]. The aim of the analysis is to compute the marginal posterior distribution of the discrete modes $p(z_{0:t}|y_{1:t})$. This distribution can be derived from the posterior distribution $p(x_{0:t}, z_{0:t}|y_{1:t})$ by standard marginalization. The posterior density satisfies the following recursion:

$$\begin{aligned} p(x_{0:t}, z_{0:t}|y_{1:t}) &= p(x_{0:t-1}, z_{0:t-1}|y_{1:t-1}) \\ &\times \frac{p(y_t|x_t, z_t)p(x_t, z_t|x_{0:t-1}, z_{0:t-1}, y_{1:t-1})}{p(y_t|y_{1:t-1})} \end{aligned} \tag{4.1}$$

This recursion involves intractable integrals in the denominator; for this reason, we must use Particle Filtering techniques.

[3]For ease of presentation, we omit $u_{1:t}$ from the arguments of the various probability distributions.

STEP 1 (Discrete modes)

- Define the number n_z of discrete modes

STEP 2 (Markov properties)

- Specify the transition matrix and the prior distribution

$$p(z_t|z_{t-1}) \in \mathbb{R}^{n_z \times n_z} \quad z_0 \sim p(z_0) \in \mathbb{R}^{n_z \times 1}$$

STEP 3 (Identification)

- Step 3a **(Designing the experimental tests)**
  - Design and implement step response test
  - Design and implement the PRBS test
- Step 3b **(Learning the deterministic model)**
  - Using the Least Squares Error algorithm compute the deterministic model : A, F, C, G
- Step 3c **(Learning the probabilistic model)**
  - Using the Expectation-Maximization algorithm compute the probabilistic model. Compute the noise matrices (B, C) and re-compute (A, F, C, G)
- Step 3d **(Validation)**
  - If acceptable error is exceeded, try different initial conditions for the *EM* algorithm, try increasing the model complexity in the *LSE* algorithm, then try another experimental data set

Figure 4.3: *JMLG* modelling algorithm

4.3.2 Objectives

Based on the definition of the inference problem in section 4.3.1, Bayesian inference for the jump Markov linear Gaussian model relies on the posterior density $p(x_{0:t}, z_{0:t}|y_{1:t})$. There are many advantages to using numerical approximation such as Particle Filtering, but also several problems. We are interested in an inference algorithm with the following features:

- *Industrial applications.*
  We want an algorithm which can help us diagnose/estimate faults/states in real domains. The inference algo-

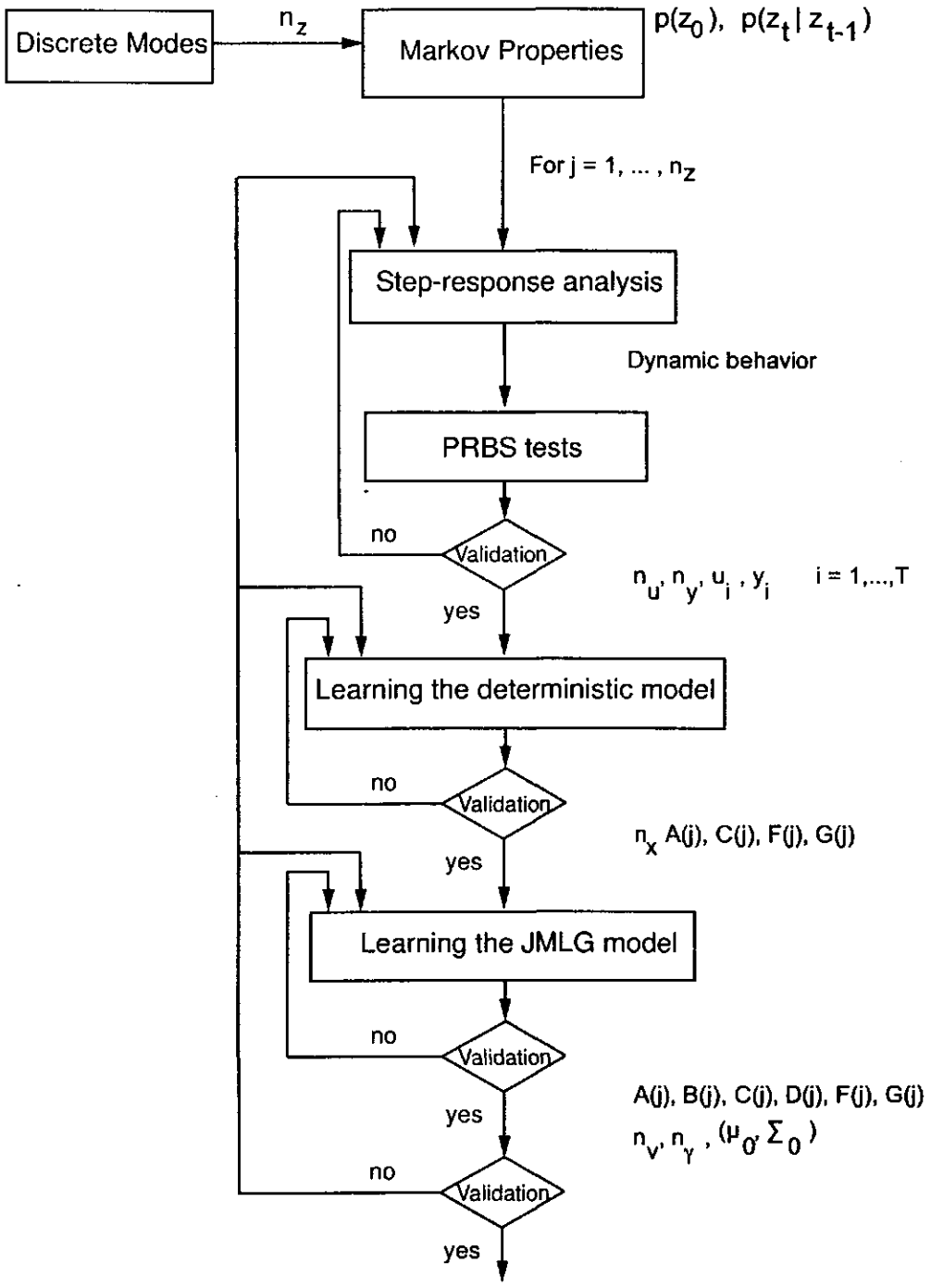

Figure 4.4: JMLG modelling procedure. This diagram shows the parameter learning procedure. After each step we show the generated information and the possible iteration loop.

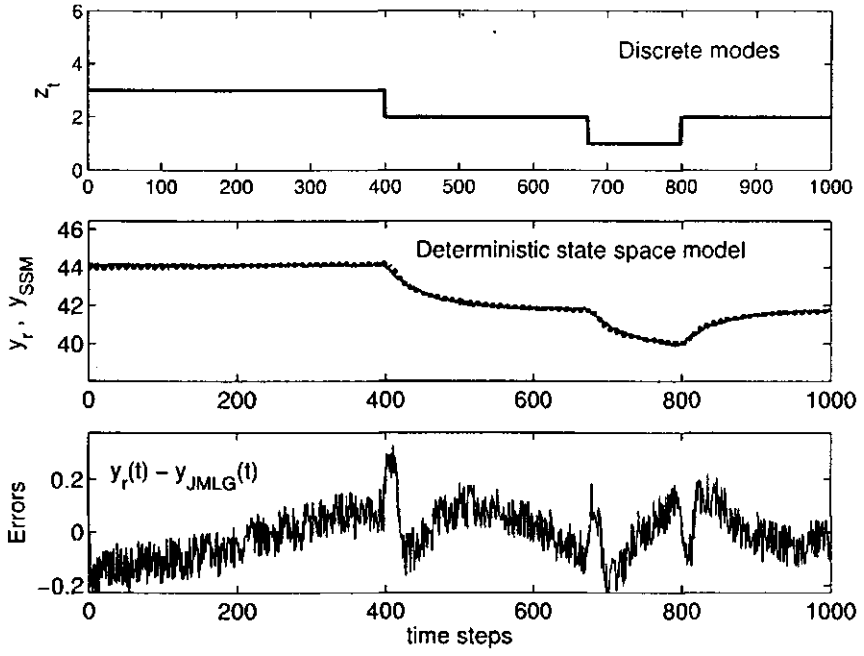

Figure 4.5: JMLG modeling results. The upper graph shows the discrete mode over time. The middle graph shows real data and the data generated by the deterministic state space model (SSM). The lower graph shows the error signal, the difference between the real data and the synthetic data generated by the jump Markov linear Gaussian model.

rithm should work efficiently in domains with different dynamic behaviours and signals.

- *Real time inference.*
  We want an algorithm to solve industrial problems. Many industrial applications rely on real time diagnosis/estimation. To satisfy this requirement, the algorithm needs to be very efficient.

- *Minimum diagnosis error.*
  The algorithm must be reliable to be part of industrial applications. Fault diagnosis using numerical approximation techniques has a special problem that can increase the diagnosis error: transitions to faulty discrete modes typically have very low probabilities. Using Particle filtering, faulty discrete modes often have no particles, and are thus erroneously considered to have zero probability. We want to improve on this.

- *Minimum variance.*
  Particle Filtering is a probabilistic approach; unlike deterministic methods, its results are given in terms of averages and variances. Nevertheless, we want to consistently obtain results with minimum variance.

4.3.3 Proposed algorithm

The jump Markov linear Gaussian model combines discrete modes (z_t), continuous states (x_t), and continuous observations (y_t, u_t). For this reason, we must work with algorithms for approximate inference. The method is the Monte Carlo Particle Filter (discussed in Chapter 3, section 3.4). Rao-Blackwellized Particle Filtering (*RBPF*)

[Akashi and Kumamoto, 1977; Doucet *et al.*, 2000b] is a Particle Filtering variant based on the Rao-Blackwell formula [Casella and Robert, 1996]. Basically, *RBPF* combines a:

- *Particle Filter* to compute the distribution of the discrete modes z_t, with a
- *Bank of Kalman Filters* to analytically compute the distribution of the continuous states x_t.

As we showed for the *Rao-Blackwellized Particle Filtering* algorithm in Chapter 3[4], by considering the factorization $p(x_{0:t}, z_{0:t}|y_{1:t}) = p(x_{0:t}|y_{1:t}, z_{0:t})p(z_{0:t}|y_{1:t})$ [Casella and Robert, 1996] it is possible to design efficient algorithms with lower variance. The density $p(x_{0:t}|y_{1:t}, z_{0:t})$ is Gaussian and can be computed analytically if we know the marginal posterior density $p(z_{0:t}|y_{1:t})$. This density satisfies the alternative recursion:

$$p(z_{0:t}|y_{1:t}) = p(z_{0:t-1}|y_{1:t-1}) \times \frac{p(y_t|y_{1:t-1}, z_{0:t})\, p(z_t|z_{t-1})}{p(y_t|y_{1:t-1})} \tag{4.2}$$

Because the analytical solution of equation (4.2) involves intractable integrals, we use a weighted set of particles $\{z_{0:t}^{(i)}, w_t^{(i)}\}_{i=1}^N$ to represent the marginal posterior $p(z_{0:t}|y_{1:t})$. The marginal density $p(x_{0:t}|y_{1:t}, z_{0:t})$ is a Gaussian mixture that can be efficiently computed with a stochastic bank of Kalman filters.

In order to deal with unexpected observations, it is possible to improve standard *RBPF* by looking one step ahead. In the *Sequential Importance Sampling* step for standard *PF* and *RBPF*, we need to have an approximation of $p(z_{1:t}|y_{1:t})$ using an importance distribution $q(z_{1:t}|y_{1:t})$ at any time t, and to be able to propagate this estimate in time without subsequently modifying the past simulated trajectories $\{z_{1:t}^{(i)}\}_{i=1}^N$. We are looking for a recursive formula, easy to implement. We can specify the importance function form as:

$$\begin{aligned} q(z_{1:t}|y_{1:t}) &= q(z_1|y_1)q(z_2|y_{1:2}, z_1)q(z_3|y_{1:3}, z_{1:2}) \cdots q(z_t|y_{1:t}, z_{1:t-1}) \\ q(z_{1:t}|y_{1:t}) &= q(z_1|y_1) \prod_{k=2}^{t} q(z_k|y_{1:k}, z_{1:k-1}) \end{aligned} \tag{4.3}$$

Equation (4.3) allows for the recursive evaluation of the importance weight w_t which is given by

$$\begin{aligned} w_t &= \frac{p(z_{1:t}|y_{1:t})}{q(z_{1:t}|y_{1:t})} \\ &= \left[\frac{p(y_t|y_{1:t-1}, z_{1:t})p(z_t|z_{t-1})}{p(y_t|y_{1:t-1})}\right] \frac{1}{q(z_t|y_{1:t}, z_{1:t-1})} \\ &\propto \frac{p(y_t|y_{1:t-1}, z_{1:t})p(z_t|z_{t-1})}{q(z_t|y_{1:t}, z_{1:t-1})} \end{aligned} \tag{4.4}$$

The proportionality in Equation (4.4) arose because we deleted the constant term $p(y_t|y_{1:t-1})$. There are many possible choices for $q(z_{1:t}|y_{1:t})$; but all must include the support of $p(z_{1:t}|y_{1:t})$. We are interested in a proposal that minimizes the variance of the importance weights. Also, this proposal must have the form shown in equation 4.3, where we consider $z_{1:t-1}$ and $y_{1:t}$ to be known information.

The minimum variance of the importance weight at time t, conditional on $y_{1:t}$ and $z_{1:t-1}$, is obtained using $p(z_t|y_{1:t}, z_{1:t-1})$ as the optimal importance distribution.

[4]We represent some previous results for completeness.

This optimal distribution allows us to compute the probability of the discrete modes $\{z_t = k\}_{k=1}^{n_z}$

$$p(z_t = k|y_{1:t}, z_{1:t-1}) = \frac{p(y_t|y_{1:t-1}, z_{1:t-1}, z_t = k)p(z_t = k|z_{t-1})}{p(y_t|y_{1:t-1}, z_{1:t-1})} \tag{4.5}$$

and the importance weights are

$$\begin{aligned} w_t \propto p(y_t|y_{1:t-1}, z_{1:t-1}) &= \sum_{k=1}^{n_z} p\big(y_t|y_{1:t-1}, z_{1:t-1}, z_t = k\big)p(z_t = k|z_{t-1}) \\ &= \sum_{k=1}^{n_z} \mathcal{N}\Big(\widehat{y}_{t|t-1}(z_t = k), \widehat{S}_t(z_t = k)\Big)p(z_t = k|z_{t-1}) \end{aligned} \tag{4.6}$$

where the parameters in equation (4.6),

- $\widehat{y}_{t|t-1} \triangleq \mathbb{E}(y_t|y_{t-1})$, and
- $\widehat{S}_t \triangleq cov(y_t|y_{1:t-1})$,

are the innovation and the prediction covariance of the observation conditional on $z_{1:t-1}$ and $z_t = k$ $\big(k \in \{1, \ldots, n_z\}\big)$. We have to compute one step of the Kalman filter in order to obtain the importance weight. Computing the importance weight requires n_z evaluations of the term $\mathcal{N}\big(\widehat{y}_{t|t-1}, \widehat{S}_t\big)$. This could be computationally expensive when the number of possible discrete modes n_z is large. However, when the number is small, say 10 or 100, we can compute the distributions in equation (4.6) analytically.

As we can see in equation (4.6), the importance weights do not depend on z_t; we are marginalizing over this variable. It is therefore possible to select particles before the sequential importance sampling step. We can choose the fittest particles at time $t-1$ using the information at time t.

These are the main ideas behind the new efficient algorithm, *look-ahead* Rao-Blackwellized Particle Filtering (*la-RBPF*) [Morales-Menéndez *et al.*, 2002; Morales-Menéndez *et al.*, 2003]. For the standard *PF* and *RBPF* discussed in Chapter 3, the importance weights depend on the sample $z_t^{(i)}$, thus not permitting selection before sampling. Selecting particles before sampling results in a richer sample set at the end of each time step. Basically, we are sampling the discrete modes directly from the posterior distribution.

Following is a summary of the basic steps for *la-RBPF*. But first, for comparison, we also show standard *RBPF* (Chapter 3 section 3.5).

Standard Rao-Blackwellized Particle Filtering

For each particle:

1. Sample a new discrete mode from the transition prior
2. Compute the prior mean and covariance
3. Update the mean and covariance for the continuous parameters
4. Weight the particles as the observation probability given the prior observation mean and covariance
5. Select the fittest particles
6. Update the sufficient statistics using one step of the Kalman recursion

Look-ahead Rao-Blackwellized Particle Filtering

For each particle:

1. Look ahead at each possible discrete mode $z^{la} \in \{1, \ldots, n_z\}$
2. Update the approximations of the continuous parameters as if we had sampled z^{la}
3. Compute the observation likelihood with these approximations
4. Compute the posterior probability of z^{la} as the new discrete mode
5. Compute the weight of each particle
6. Select the fittest particles
7. Sample a discrete mode directly from the posterior probability distribution
8. Update the sufficient statistics using one step of the Kalman recursion

The key steps in *look-ahead Rao-Blackwellized Particle Filtering* versus standard *RBPF* are the order of sampling (7) and selection (6) steps, and sampling from the posterior. The pseudo-code for *la-RBPF* is shown in Figure 4.6.

Because we are sampling the discrete modes from the true posterior probability distribution, very low prior probabilities (i.e. fault probabilities) do not represent a big problem. The observation likelihood can capture the evidence over time, so if a faulty discrete mode appears, it is identified eventually. We may have to wait for enough evidence to accumulate in order to overcome the low prior probability, but we can detect it.

Figure 4.7 shows a graphical representation of the standard Particle Filter, consisting of two steps (sequential importance sampling and the selection or resampling step, Chapter 3 section 3.4). We want to emphasize the following events:

- A set of 10 resampled (selected) particles exists at time *t-1*.
- A new set of 10 particles is sampled using the transition prior at time *t-1*.
- The importance weights are computed for each particle at time *t-1*.
- A new set of 10 particles is resampled (selected) according to the weights (previous step) at time *t*.

Now, we illustrate the same events as they occur in the *la-RBPF* algorithm. For *la-RBPF* the events are in the following order:

- A set of 10 resampled (selected) particles exists at time *t-1*.
- The importance weights are computed for each particle at time *t-1*.
- A new set of 10 particles is resampled (selected) according to the weights (previous step) at time *t-1*. We are choosing the fittest particles at time *t-1* using the information at time *t*.
- A new set of 10 particles is sampled at time *t*.

As we can see in Figure 4.8, selecting particles before sampling results in a richer sample set at the end of each time step, and we are more likely to be able to deal with changing filtering distributions.

Kalman prediction step

- For i=1, ..., N, and for $z_t = 1, \ldots, n_z$ compute

$$\widehat{\mu}^{(i)}_{t|t-1}(z_t), \;\; \widehat{\Sigma}^{(i)}_{t|t-1}(z_t), \;\; \widehat{y}^{(i)}_{t|t-1}(z_t), \;\; \widehat{S}^{(i)}_t(z_t)$$

- For i=1 , ..., N , evaluate and normalize the importance weights

$$\begin{aligned} w^{(i)}_t &= p(y_t|y_{1:t-1}, z^{(i)}_{0:t-1}) \\ &= \sum_{z_t=1}^{n_z} \mathcal{N}(\widehat{y}^{(i)}_{t|t-1}(z_t), \widehat{S}^{(i)}_t(z_t)) p(z_t|z^{(i)}_{t-1}) \end{aligned}$$

Selection step

- Multiply/Discard particles $\left\{\widehat{\mu}^{(i)}_{t-1}, \widehat{\Sigma}^{(i)}_{t-1}, \widehat{z}^{(i)}_{0:t-1}\right\}_{i=1}^{N}$ with respect to high/low importance weights $w^{(i)}_t$ to obtain N particles $\left\{\mu^{(i)}_{t-1}, \Sigma^{(i)}_{t-1}, z^{(i)}_{0:t-1}\right\}_{i=1}^{N}$.

Sequential importance sampling step

- *Kalman prediction.* For i=1, ..., N, and for $z_t = 1, \ldots, n_z$ using the resampled information, re-compute

$$\widehat{\mu}^{(i)}_{t|t-1}(z_t), \;\; \widehat{\Sigma}^{(i)}_{t|t-1}(z_t), \;\; \widehat{y}^{(i)}_{t|t-1}(z_t), \;\; \widehat{S}^{(i)}_t(z_t)$$

- For $z_t = 1, \ldots, n_z$ compute

$$p(z_t|z^{(i)}_{0:t-1}, y_{1:t}) \propto \mathcal{N}(\widehat{y}^{(i)}_{t|t-1}(z_t), \widehat{S}^{(i)}_t(z_t)) p(z_t|z^{(i)}_{t-1})$$

- *Sampling step*

$$z^{(i)}_t \sim p(z_t|z^{(i)}_{0:t-1}, y_{1:t})$$

Updating step

- For i=1 , ..., N, use one step of the Kalman recursion to compute the sufficient statistics $\left\{\mu^{(i)}_t, \Sigma^{(i)}_t\right\}$ given $\left\{\widehat{\mu}_{t|t-1}(z^{(i)}_t), \widehat{\Sigma}_{t|t-1}(z^{(i)}_t)\right\}$.

Figure 4.6: *la-RBPF* algorithm at time t. The algorithm uses an optimal proposal distribution. It also selects particles from time $t-1$ using the information at time t. $\mu_{t|t-1} \triangleq \mathbb{E}(x_t|y_{1:t-1})$, $\mu_t \triangleq \mathbb{E}(x_t|y_{1:t})$, $y_{t|t-1} \triangleq \mathbb{E}(y_t|y_{1:t-1})$, $\Sigma_{t|t-1} \triangleq cov(x_t|y_{1:t-1})$, $\Sigma_t \triangleq cov(x_t|y_{1:t})$ and $S_t \triangleq cov(y_t|y_{1:t-1})$.

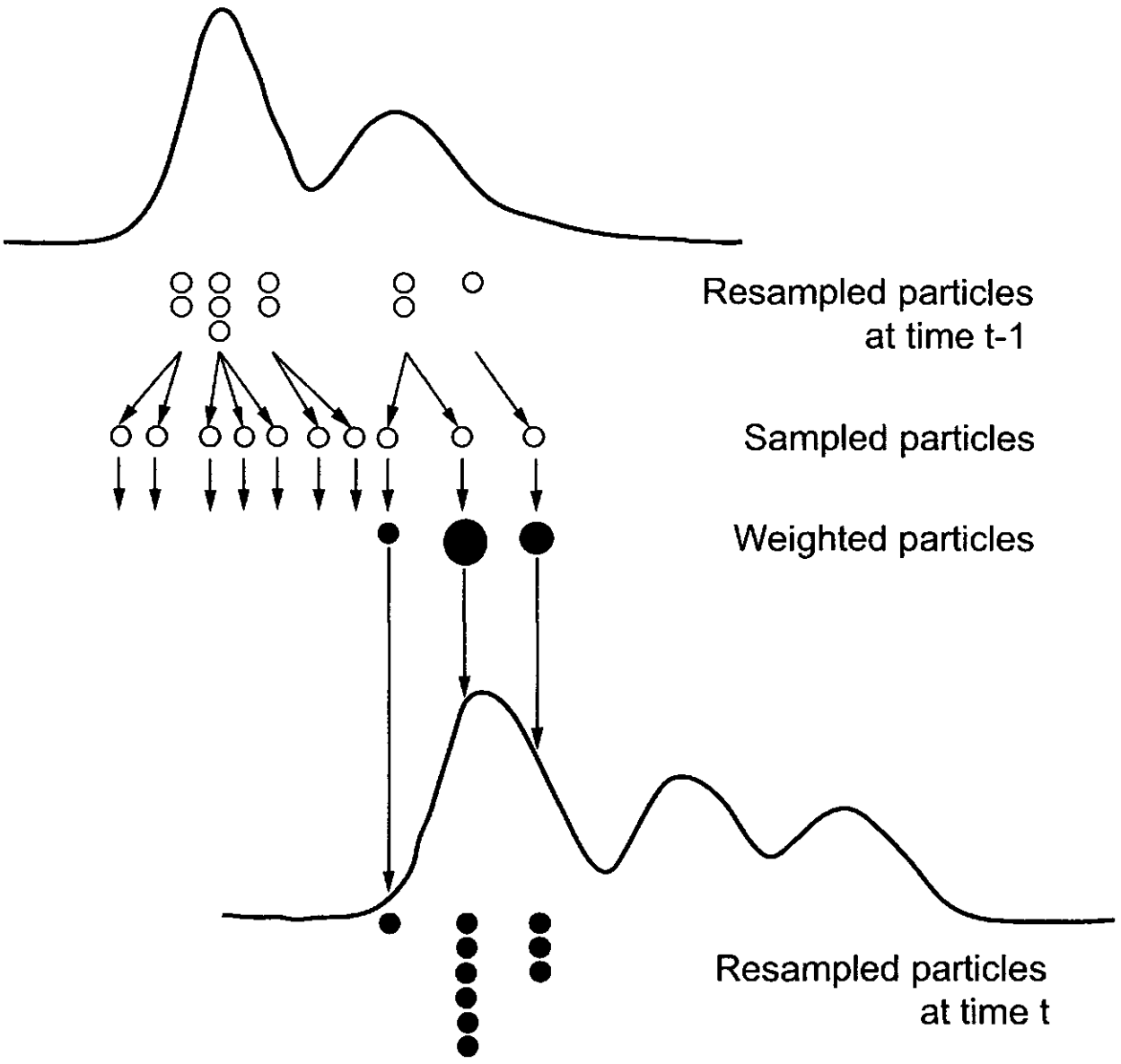

Figure 4.7: Graphical representation, Standard Particle Filtering. Starting with the resampled particles at time *t-1*, a new set of particles is proposed at time *t*. We compute the importance weight of each particle. Finally, we select the fittest particles according to their weights. Note the filter has failed to track the modes appearing at the right of the filtering posterior distribution at time *t*.

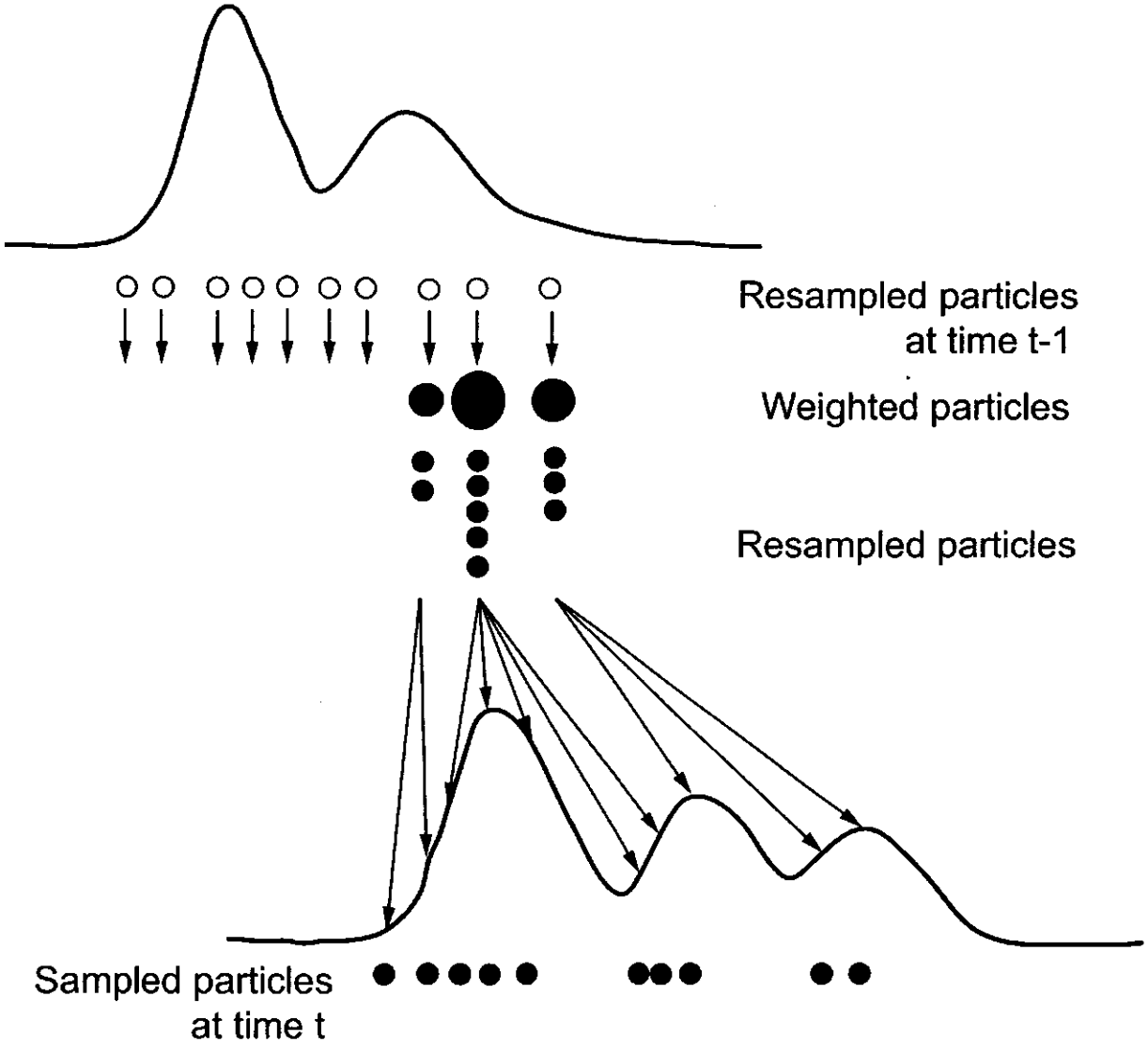

Figure 4.8: Graphical representation, look-ahead Rao-Blackwellized Particle Filtering algorithm. First we compute the importance weights. After resampling according to these weights, we propose new particles at time t. With this algorithm, we are more likely to be able to track all modes of the changing filtering distribution at time t.

La-RBPF Convergence

The *RBPF* convergence proof of [Doucet *et al.*, 2000a] also applies to *la-RBPF*; the main difference between these algorithms is the proposal distribution. The *Rao-Blackwellization* sampling scheme used in *la-RBPF* is widely demonstrated in [Casella and Robert, 1996], and a complete explanation is included in Appendix C section C.2. Also, for completeness, we include the following convergence proof.

Let B $(\mathbb{R}^n)$ be the space of bounded, Boreal measurable functions on $\mathbb{R}^n$. $\| \mathbf{g} \| \triangleq \sup_{x \in \mathbb{R}^n} |\mathbf{g}(\mathbf{x})|$. The following theorem was given in [Crisan and Doucet, 2000]. In addition, [Crisan and Doucet, 2002] presents a survey of convergence results on Particle Filtering methods.

Theorem 1 *If the importance weights w_t are upper bounded and if we use one of the selection schemes described in Appendix B, section B.2.2, then, for all $t \leq 0$, there exists c_t independent of N such that for any $g_t \in B\left((\mathbb{R}^{n_z})^{t+1}\right)$*

$$\mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_t(\mathbf{z}_{0:t}^{(i)}) - \int \mathbf{g}_t(\mathbf{z}_{0:t})p(\mathbf{z}_{0:t}|\mathbf{y}_{1:t})d\mathbf{z}_{0:t}\right)^2\right] \leq c_t \frac{\| \mathbf{g}_t \|^2}{N} \tag{4.7}$$

where the expectation is taken with respect to the randomness introduced by the Particle Filtering algorithm (such as *la-RBPF*). The convergence of this general Particle Filter is guaranteed and the convergence rate is independent of the dimension of the state space[5].

4.4 Main Results

We discuss our results in three main areas: *industrial applications*, *JMLG* model and *la-RBPF* algorithm. This is a brief summary; most of these results are discussed and analyzed in detail in Chapter 7, supported by experiments. *La-RBPF* algorithm represents our main contribution in this research.

4.4.1 Industrial Applications

To the best of our knowledge and belief there is no published literature in this field. Specifically, there are no papers which exploit this principled probabilistic technique, Monte Carlo Particle Filtering, for estimation/diagnosis in technical processes such as:

- Shell-and-tube heat exchanger
- Industrial dryer
- Level-tank.

In the last five years, many important papers have appeared using similar tools, but their results have been based on simulated equations, simplified models, etc. Nevertheless these papers show impressive results and make very important contributions in many fields. Another important group of papers have appeared in robotics, vision, etc., making important contributions in this area.

Our research was based mainly on real-life applications in industrial or pilot processes. We believe that the practical nature of this research is one of its most important contributions. Fault Detection and Isolation (*FDI*) systems are the classical approach to tackling the diagnosis/estimation problem in the control engineering community, and

[5] c_t usually increases exponentially over time.

there is a lot of research in this area (Chapter 2); however, our proposal is innovative in this practical field. We believe it represents a potential tool for other applications due to its on-line properties.

We also tested our algorithm in the robotics field using real data with excellent results, [de Freitas *et al.*, 2003].

4.4.2 JMLG model

We got excellent results representing continuous dynamic processes using the jump Markov linear Gaussian (*JMLG*) model, see Figure 4.5.

Depending on the inference task we are interested in, the model designer requires certain process knowledge:

- *Fault diagnosis.*
  If we want to use the *JMLG* model to diagnose faults, we must identify all the different possible faulty points (and their combinations). The difficult task here is designing the transition matrix and prior distribution. Usually faults are unpredictable and difficult to analyze.

- *State estimation.*
  If we are interested in state estimation, we must identify the different operating conditions the process can be in. Nonlinearity could appear; we need to break nonlinear states into several pieces that are approximately linear Gaussian.

A lot of work was required for the experimental tests, however the *JMLG* model always performed well. We tested our models in processes with fast and slow response, high and low noise, one and more dimensions for the continuous state variables, and four or more discrete modes. In every case, the results were excellent.

The *JMLG* model represents a simple framework for modelling nonlinear processes or processes with different operating conditions. Given this model, the diagnosis/estimation task is relatively straightforward.

The proposed learning procedure gave good results; however, there are some possible practical problems. It can be difficult to implement persistent input signals for fault states, because it is difficult to reproduce faulty situations. Indeed, it is difficult to identify and implement all the possible faulty conditions. Nevertheless, we can consider the most common ones and design a practical system. It is important to note that this problem also exists in other model-based approaches.

4.4.3 Look-ahead Rao-Blackwellized Particle Filtering

We tested the three Particle Filtering algorithms described in this chapter and the previous chapter 3. Representative results are shown in Figure 4.9. The left graph shows diagnosis error versus number of particles, while the right graph shows diagnosis error versus computing time per time step. Diagnosis error is the percentage of time steps during which the discrete mode was not identified properly. The *Maximum A Posteriori* (*MAP*) was used to define the most probable discrete mode over time.

- *Diagnosis error versus number of particles.*
  The proposed algorithm *la-RBPF* gives a very low diagnosis error per number of particles. It works significantly better than standard *PF* and *RBPF*.

- *Diagnosis error versus computing time.*
  la-RBPF also gives a very low diagnosis error per unit of computing time, despite its greater computational expense per particle compared with standard *PF* and *RBPF*. However, if the number of discrete modes grows

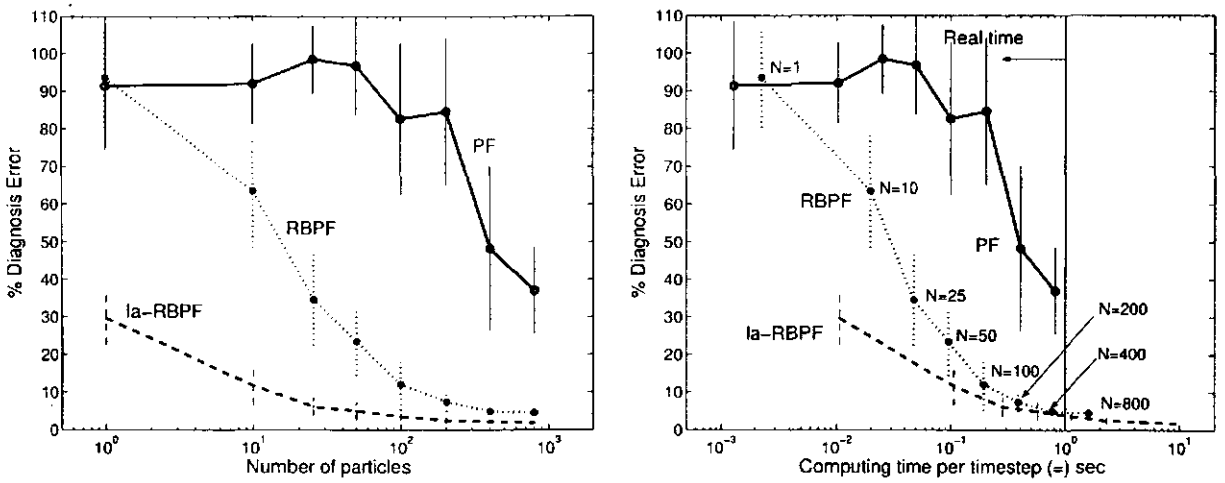

Figure 4.9: Representative diagnosis error performance. The left graph shows diagnosis error versus number of particles for the three Particle Filtering algorithms. The right graph shows diagnosis error versus computing time per time step (only, for the *RBPF* line the number of particles are shown over each data point, same number of particles are associate with the other lines.) *la-RBPF* outperforms both *PF* and *RBPF*, even considering its greater computational expense per particle. This demonstrates its usefulness for real-time applications. Each data point shows the average and standard deviation computed from 25 independent runs. Each run has 1,000 time steps.

too high, *la-RBPF* may not be able to work in real-time. There are some possible solutions for this problem: parallel computing or more efficient languages[6].

- *Time response*
  la-RBPF requires less time to diagnose/estimate a discrete mode.

- *Very low probabilities.*
  Faulty conditions usually have very low probabilities. Standard numerical approximations have trouble with this situation because a very small number of particles are assigned to a faulty discrete mode, despite the observations. However, *la-RBPF* samples the possible discrete modes from their true posterior distribution, capturing evidence of faulty conditions and allowing them to be identified.

- *Variance.*
  la-RBPF gives lower variance than standard *PF* and *RBPF* per number of particles. This advantage, based on the Rao-Blackwell formula, grows as the number of particles is increased.

- *Real time applications.*
  la-RBPF generated better results than standard *PF* and *RBPF* in real-time. However, if the number of discrete modes or the model complexity (n_x, n_y, n_u) grows, *la-RBPF* may have to work off-line.

- *Noise environment.*
  If the signals have high noise levels, looking ahead a single time-step does not tell you much about the next possible discrete mode, especially at high sampling rates. In this case the look-ahead is just extra work for no

[6]The algorithms tested were written in Matlab version 6.0.

benefit. Often, however, we can apply a very simple moving-average filter to the data, restoring *la-RBPF*'s advantage.

4.5 Summary

In the first part of this chapter, the modelling problem was defined. An algorithm for learning the *Jump Markov Linear Gaussian* model's parameters was proposed. This algorithm has two iterative stages. In the first stage, the *Least Squares Estimation* method is used to compute some parameters, i.e. the deterministic state space representation of each continuous dynamic model. In the second stage, the deterministic state space representation is used as an initial approximation for the *Expectation-Maximization* method, and the full parameters of the *Jump Markov Linear Gaussian* model are calculated. The proposed algorithm was tested in many real life applications, and the results are sufficiently good. Specifically, for our diagnosis/estimation purposes the results are excellent. There are many opportunities to improvement the proposed algorithm, as many time-consuming steps were manually implemented. Moreover, some crucial information, such as the number of discrete modes and their transition matrix, must be engineered using process knowledge.

For the inference task, we presented our main contribution the *look-ahead Rao-Blackwellized Particle Filtering* algorithm in the second part of this chapter. *la-RBPF* is a Particle Filtering variant that represents the continuous variables in a compact manner and reduces the number of particles required. *la-RBPF* samples the discrete modes directly from the posterior probability distribution and selects the fittest particles before the transition step. These two main changes make *la-RBPF* a more efficient algorithm. Results for some industrial processes showed low diagnosis error with very low variance even for faulty discrete modes (characterized by very low prior probabilities). *la-RBPF* can be computationally expensive; specifically, if the number of discrete modes grows, it may be limited to off-line applications.

Chapter 5

Experimental implementation

5.1 Introduction

We tested the *Particle Filtering* and *Rao-Blackwellized Particle Filtering* algorithms described in Chapter 3, and the new *look-ahead Rao-Blackwellized Particle Filtering* algorithm presented in Chapter 4, with four real-world systems[1]:

- industrial dryer
- level-tank
- industrial heat exchanger
- mobile robot

Each domain has different characteristics that we want to analyze and exploit, such as non-linear dynamics, noisy signals, different kinds of input/output variables, etc. Some domains have industrial characteristics, and their results are therefore 100 % transferable to similar systems; other domains have pilot-plant characteristics only. However, all of them have industrial instrumentation. The main results to show for each domain relate to:

- Jump Markov Linear Gaussian (JMLG) modelling
- Inference performance of the Particle Filtering algorithms

In this chapter, we give the main results; however, detailed discussions are deferred until Chapter 7. Each domain is described in four basic sections:

1. *Description.* A brief description of the process and our motivation for choosing it. We also define the discrete modes that can occur.
2. *Modelling.* The main modelling results, e.g. the *JMLG* parameters.
3. *Diagnosis/estimation tests.* The transition matrix, initial distribution, and random sequence chosen. Basically, the procedure to generate the experimental data set. We also validate the *JMLG* model.

[1] We thank Francisco Calleja for his support during the experimental tests at ITESM, campus Monterrey.

4. *Results.* Plots that show diagnosis error versus number of particles and computing time. For some domains we show other interesting graphical results.

For the first domain, the industrial dryer, we describe each step in detail. For the other domains we give only the final results.

Some domains are suitable for fault detection and diagnosis, whereas others are appropriate for state estimation only. We do not focus on that distinction here.

As our goal is to perform diagnosis/estimation for real time applications, we use 1 second as our reference sampling rate. This is typical of most of the domains we are testing; however, the sampling rate is domain depended.

In order to test the *jump Markov linear Gaussian* model and the *Particle Filtering* inference algorithms in more domains, having different characteristics, we developed a simulator. This also allowed us to increase the complexity of our domains. Several simulations are shown in Chapter 6, in which we manipulated features such as the number of discrete modes and the nonlinear regimes.

5.2 Industrial dryer

To test our inference algorithms, we started with a common real-world process. An industrial dryer was adapted for experimental purposes, so that faults points could be repeatedly implemented.

5.2.1 Description

The industrial dryer is a thermal process that converts electricity to heat. The dryer is able to control the exit air temperature by changing its shooting angle. It has digital communication with a personal computer using a data acquisition system, [Valles *et al.*, 1998]. The generated faults were implemented using:

- fan speed: low/high
- fan grill: closed/opened
- dryer exit vent: clear/obstructed

Figure 5.1 shows a picture of this process. Figure 5.2 shows a basic diagram with the faulty points. Four possible experimental conditions can be implemented in the dryer; details are shown in Table 5.1 [Morales *et al.*, 2001].

Table 5.1: Industrial dryer. Operating Conditions.

z_t	Model name	fan speed	dryer grill	exit vent
1	Normal operation	low	opened	clear
2	Faulty fan	high	opened	clear
3	Faulty grill	low	closed	clear
4	Faulty fan and grill	high	closed	clear

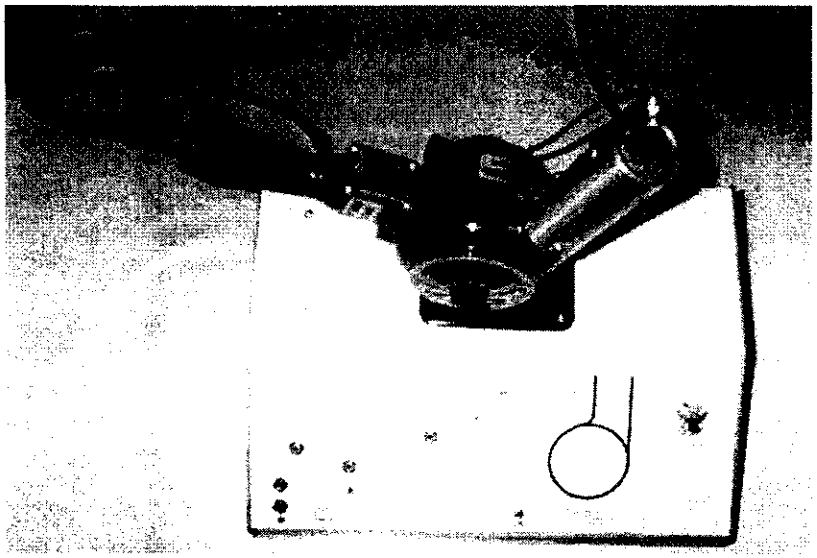

Figure 5.1: Industrial dryer. This dryer uses a motor-driven fan and a heating coil to transform electrical energy into convective heat.

5.2.2 Modelling

Parametric identification was carried out based on the Least Squares Estimation algorithm [Ljung, 1987]. Experimental step changes were applied by altering the input signal from 15 % to 25 % and vice versa using 1 sec as the sampling time. The transient response follows a first order plus dead time (FOPDT) model, equation (5.1) [Smith and Corripio, 1997].

$$Gp(s) = \frac{Y(s)}{U(s)} = \frac{Ke^{-\theta' s}}{\tau s + 1} \tag{5.1}$$

where:

- $Gp(s)$ represents the transfer function
- $Y(s)$ is the output signal, e.g. temperature
- $U(s)$ is the input signal, e.g. current percentage
- K is the process gain, e.g. $\frac{\Delta\ temperature}{\Delta\ current}$
- τ is the process lag constant, in seconds
- θ' is the process dead time, in seconds
- s is the Laplace Transform operator

Table 5.2 shows the experimental FOPDT models. u_{ss} and y_{ss} represent the input and output steady states, respectively.

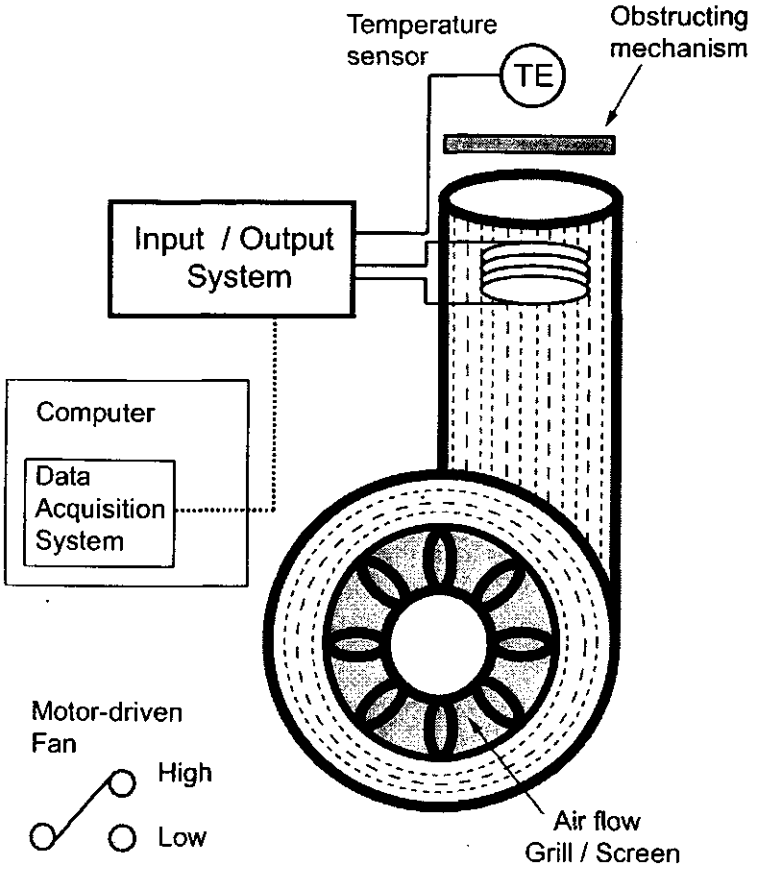

Figure 5.2: Industrial dryer (diagram). Using only measurements taken with the temperature sensor, we want to diagnose on-line the most probable faulty point at each time step.

Table 5.2: Industrial dryer. Experimental FOPDT models.

z_t	Model name	u_{ss}	y_{ss}	K	τ	θ'
1	Normal operation	15	50.52	1.94	30.45	0
2	Faulty fan	15	44.16	1.44	25.10	0
3	Faulty grill	15	54.76	2.34	41.68	0
4	Faulty fan and grill	15	46.99	1.61	32.96	0

Using the inverse Laplace transform, we get the differential equations for each model. After that we can obtain the deterministic discrete state space representation[2], choosing a sampling rate of 1 sec. Next, having the deterministic model[3], we can obtain the stochastic model. We determine the process and measurement noise matrices ($B(\cdot)$ and $D(\cdot)$) and refine the deterministic matrices ($A(\cdot)$, $C(\cdot)$, $F(\cdot)$, $G(\cdot)$) using the *Expectation-Maximization* (*EM*) algorithm [Ghahramani and Hinton, 1996]. The deterministic model is used as a good initial approximation for the *EM* algorithm in order to avoid convergence to local maxima . Equations (5.2-5.3) represent the *JMLG* model. The matrices and initial state conditions for this application are shown in Table 5.3.

$$x_{t+1} = A(z_{t+1})x_t + B(z_{t+1})\gamma_{t+1} + F(z_{t+1})u_{t+1} \tag{5.2}$$

$$y_t = C(z_t)x_t + D(z_t)v_t + G(z_t)u_t \tag{5.3}$$

Table 5.3: Industrial dryer. JMLG parameters.

z_t	$x_0(z_t)$	$A(z_t)$	$B(z_t)$	$C(z_t)$	$D(z_t)$	$F(z_t)$	$G(z_t)$
1	50.52	0.9676	0.05	1	0.05	1.6321	0
2	44.16	0.9609	0.05	1	0.05	1.7247	0
3	54.76	0.9762	0.05	1	0.05	1.2981	0
4	46.99	0.9701	0.05	1	0.05	1.4042	0

Some intermediate results of this modelling procedure (for the industrial heat exchanger) are given in Appendix D section D.3.1.

5.2.3 Diagnosis/estimation tests

More than 20 random sequences were physically implemented in this domain. These sequences were designed using the transition matrix and prior probabilities shown in equation (5.4). Figure 5.3 shows a graphical representation of this transition matrix.

$$P(z_t|z_{t-1}) = \begin{bmatrix} 0.994 & 0.002 & 0.002 & 0.002 \\ 0.002 & 0.994 & 0.002 & 0.002 \\ 0.002 & 0.002 & 0.994 & 0.002 \\ 0.002 & 0.002 & 0.002 & 0.994 \end{bmatrix} \quad P(z_0) = \begin{bmatrix} 0.988 & 0.004 & 0.004 & 0.004 \end{bmatrix} \tag{5.4}$$

Table 5.4 shows a typical random sequence[4], consisting of 6 time intervals. In each interval a condition (faulty or normal) was physically generated. During the first time interval (1-70 time steps), the dryer was working under normal operating conditions. In the second interval (71-238), a faulty fan condition was introduced, etc. We obtained data points for 1,000 time steps; air temperature was the observable variable $\{y_t\}$ for each faulty condition $\{z_t\}$.

Also, following the same random sequence in Table 5.4, and using the *JMLG* model parameters, we simulated the *JMLG* model behaviour. Figure 5.4 compares the real and simulated data for the industrial dryer[5]. The upper

[2]Also, using the z-Transform [Ogata, 1995], we can convert Gp(s) and then get the state space representation.

[3]We thank Jorge Limón-Robles and Ricardo Ramírez-Mendoza (ITESM) for their excellent discussions regarding state space representation.

[4]All the results in this section refer to this sequence.

[5]For clarity in these comparisons, we always omit the noise matrices in the *JMLG* model.

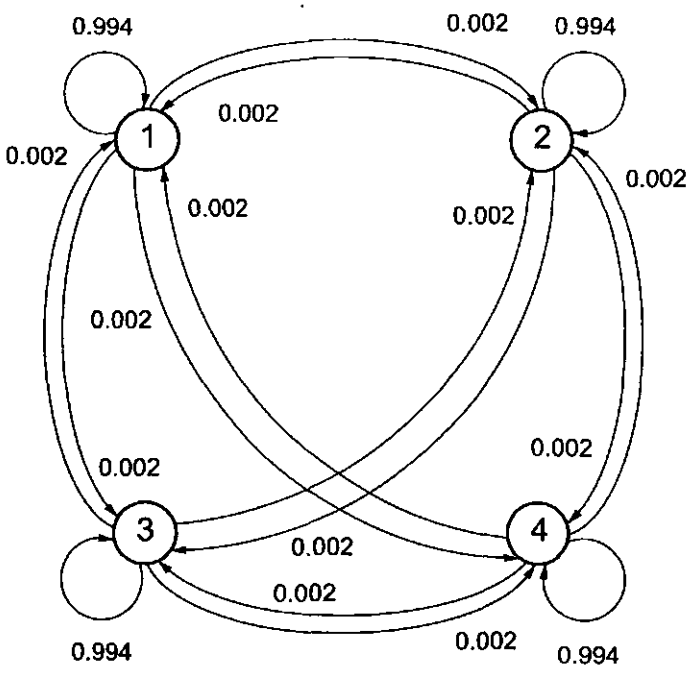

Figure 5.3: Industrial dryer. Transition matrix

graph shows how the discrete mode z_t changes over time; the lower graph shows how the air temperature (real and simulated data) changes for each discrete mode.

Three more random sequences for the industrial dryer are given in Appendix D section D.1.1, in order to further validate the *JMLG* model's performance.

Table 5.4: Industrial dryer. Random sequence No. 1.

Step	Interval	z_t	Model name	Fan speed	Dryer grill	Exit vent
1	(1 , 70)	1	Normal operation	low	opened	clear
2	(71 , 238)	2	Faulty fan	high	opened	clear
3	(239, 365)	3	Faulty grill	low	closed	clear
4	(366, 513)	1	Normal operation	low	opened	clear
5	(514, 705)	3	Faulty grill	low	closed	clear
6	(706, 1,000)	2	Faulty fan	high	opened	clear

5.2.4 Results

Given the real observations (air temperature in this domain) and the discrete modes over time, we tested the three Particle Filtering algorithms. Table 5.5 shows the performance of each algorithm using random sequence No. 1. Each algorithm was tested with different numbers of particles, N. We define *diagnosis error* as the percentage of time steps during which the discrete mode was not identified properly. We use the *Maximum A Posteriori* (*MAP*) in

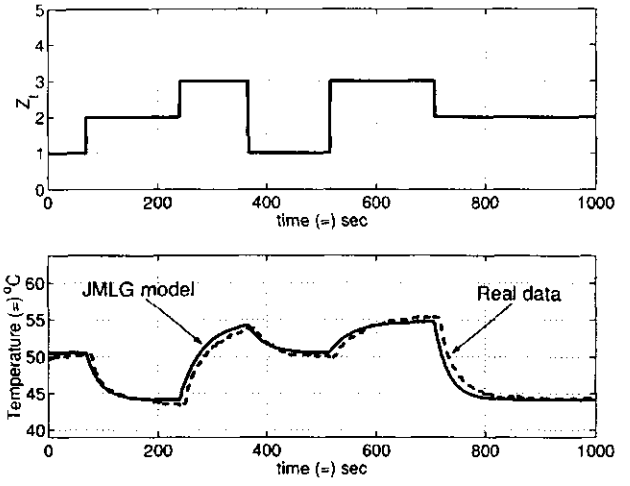

Figure 5.4: Industrial dryer. Random sequence No. 1. The upper graph shows the discrete modes over time. The lower graph compares the real temperature and the synthetic data obtained using the *jump Markov linear Gaussian* model for the same discrete mode changes.

order to define the most probable discrete mode over time.

$$Diagnosis\ error = \frac{Time\ steps\ having\ discrete\ mode\ wrongly\ identified}{Total\ number\ of\ time\ steps} \times 100\%$$

A 10 % diagnosis error for 1,000 data points means that the algorithm identified the discrete mode for 900 time steps correctly, and 100 incorrectly. Because Particle Filtering is a stochastic algorithm, we performed 25 independent runs of 1,000 time steps each. With this data we computed the mean and standard deviation (SD) of the diagnosis error for each algorithm for different numbers of particles. These numerical results are shown in Table 5.5.

In Figure 5.5 we show the same results graphically. In the left graph, we plot the diagnosis error mean versus the number of particles; we include the standard deviation around each mean data point.

Since RBPF and la-RBPF require more computing time per time step, a better comparison is diagnosis error mean versus computing time per time step. The right graph in Figure 5.5 shows this. Again, we include the standard deviation around the mean data points, and we draw a vertical line at 1 sec to separate on-line and off-line diagnosis. Note that the exact location of this boundary depends on the application.

There is a baseline error rate resulting from human error in timing the discrete mode changes. These changes were manually implemented in all real (non-simulated) domains.

Figures (5.6-5.7) show the Maximum A Posteriori (*MAP*) estimate over time. Each graph shows the true discrete mode and the MAP estimate for each Particle Filtering algorithm. The overall diagnosis error for each algorithm is also given; note that this represents a single run, not an average. The upper plots for each graph correspond to standard Particle Filtering, the middle plots to *RBPF*, and the lower plots to *look-ahead RBPF*.

The left graphs in Figure 5.6 were generated using 50 particles, while the right graphs used 100. In Figure 5.7 we used 200 and 400 particles, respectively. As we can see, standard Particle Filtering improves as the number of particles grows; however, for *RBPF* and *la-RBPF* the improvement is small because fewer particles are required for better estimates. (Recall there is a baseline error rate resulting from human error.)

More results are included in Appendix D section D.1.2.

Table 5.5: Industrial dryer. Diagnosis error for random sequence No. 1.

Particles	% Diagnosis error, mean			% Diagnosis error, SD		
N	PF	RBPF	la-RBPF	PF	RBPF	la-RBPF
1	75.13	75.61	26.02	10.20	14.17	7.79
4	77.81	57.69	18.82	8.38	9.98	5.74
10	68.35	39.07	13.55	16.11	12.03	5.62
25	50.50	21.61	9.08	11.04	7.18	3.49
50	40.06	14.78	7.76	9.99	5.06	2.16
100	24.20	10.20	7.28	7.16	3.77	1.15
200	17.02	7.99	6.87	5.03	0.55	0.29
400	13.77	7.50	6.79	3.97	0.30	0.18
800	13.99	7.53	6.79	1.93	0.19	0.09
1600	13.07	7.42	6.75	2.14	0.15	0.10

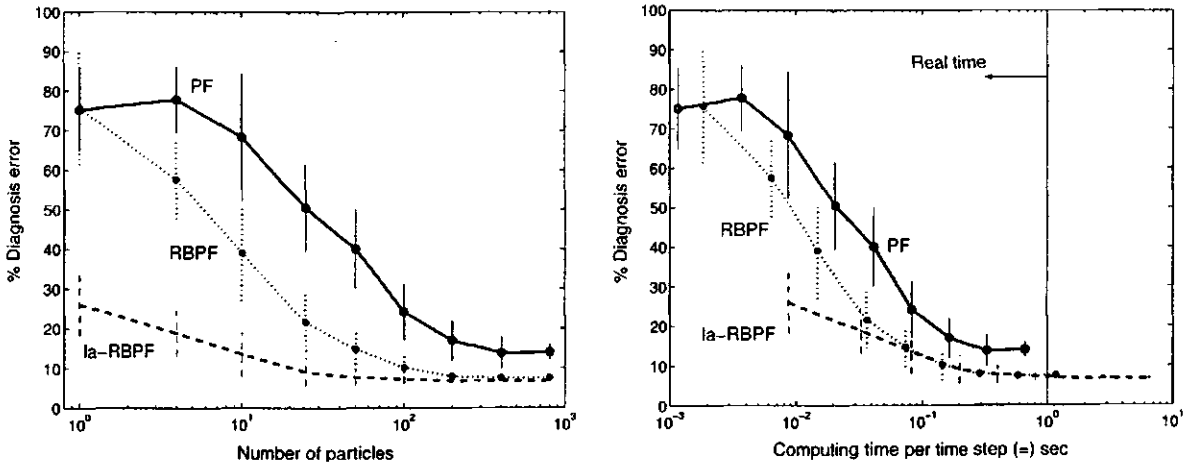

Figure 5.5: Industrial dryer. Diagnosis error for random sequence No. 1. The left graph shows diagnosis error versus number of particles, while the right graph shows diagnosis error versus computing time per time step. A vertical line appears where the computing time is 1 sec. To the left of this boundary, the number of particles is low enough to allow the Particle Filtering algorithms to be implemented on line.

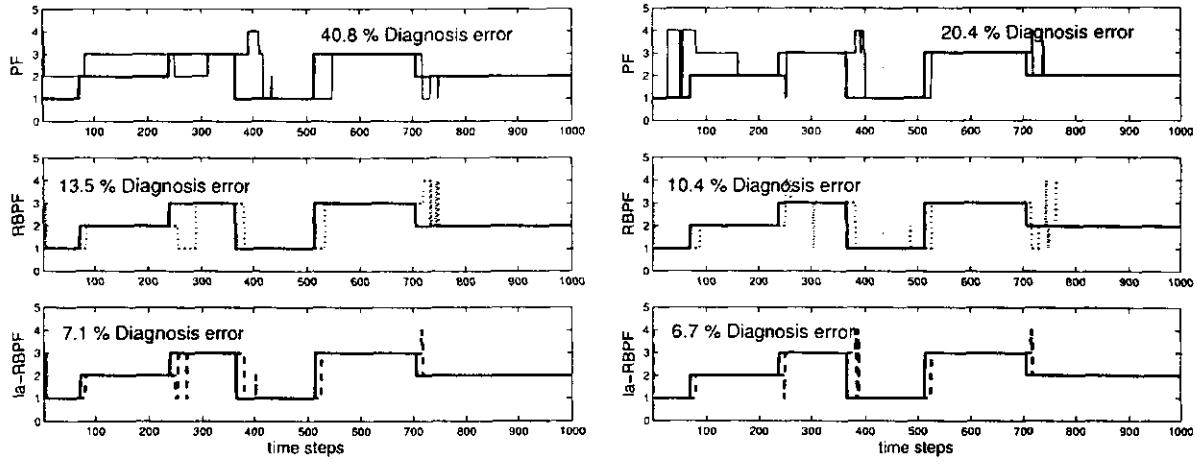

Figure 5.6: Industrial dryer. MAP estimation for random sequence No. 1. Each plot shows the true discrete mode and the MAP estimate generated by each *PF* algorithm. The overall diagnosis error for each algorithm is also given. We used 50 particles for the left plots and 100 particles for the right ones.

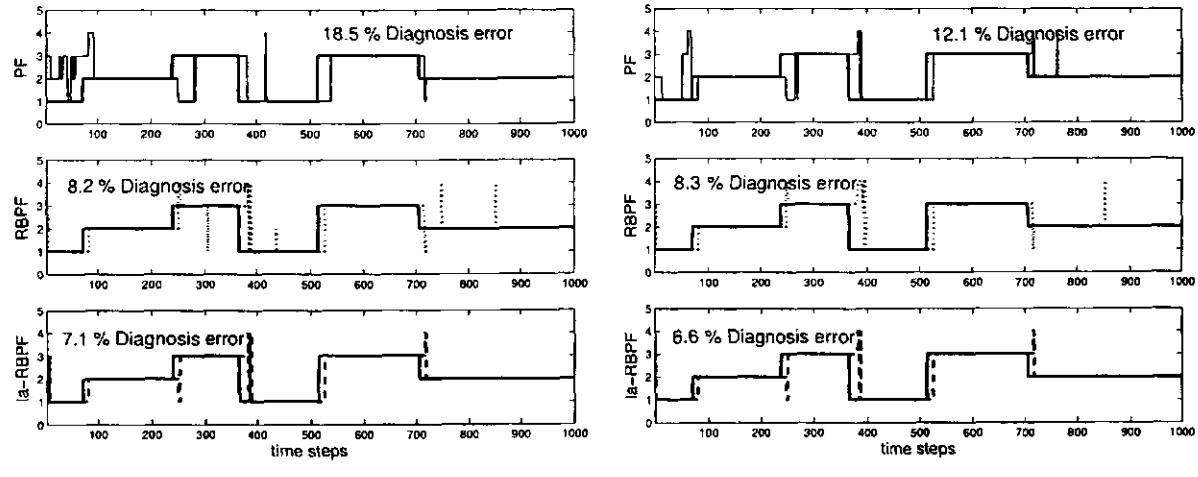

Figure 5.7: Industrial dryer. MAP estimation for random sequence No. 1. Each plot shows the true discrete mode and the MAP estimate generated by each *PF* algorithm. The overall diagnosis error for each algorithm is also given. We used 200 particles for the left plots and 400 particles for the right ones.

Variation in the transition matrix

To test the robustness of our Particle Filtering algorithms, we created some variations in the transition matrix probabilities, Equation (5.4). Despite these changes, the diagnosis error remained the same; see Tables (D.4-D.5)[6] in Appendix D, section D.1.2.

These results were expected because Particle Filtering obeys Bayes' theorem. Basically, prior distribution and likelihood functions relating to the observations are combined in order to get the posterior distribution. All inference is based on the posterior distribution. The posterior distribution is proportional to the likelihood times the prior, and is updated as data become available. Inference is implemented on-line where the observations (evidence) arrive sequentially; the prior distribution becomes less important over time.[7]

5.3 Level tank

The level tank is a widely studied system because it can represent the dynamic behaviour of many industrial processes, such as a boiler drum, part of a distillation column, part of an evaporator, etc.

5.3.1 Description

The pilot plant that we used is shown in Figure 5.8. See Figure 5.9 and Table 5.6 for a simple yet complete description of the industrial instrumentation in this experimental system. The instruments have standard analog (4-20 mA) and digital communication with a Honeywell UDC 6300 Controller (not shown), which in turn has digital communication with a computer using the Honeywell LeaderLine PC software.

A manual bypass valve, V_1, is used to physically implement some faulty feed-water pump behaviours. There is another manual valve, V_2, at the output pipe. There are more accessories and devices (e.g. air filters, pumps, solenoids, etc.), but they were omitted from the diagrams for simplicity.

We designed two sets of discrete modes to test this domain. Each test is analyzed separately in the following sections.

- 4 discrete modes
- 5 discrete modes

Table 5.6: Level tank. Instrumentation.

Tag-name	Functional name	Description	Units
FT-100	Flow sensor/transmitter	Input flow measurement	%
FV-100	Flow valve	Input flow control valve	%
LT-100	Level sensor/transmitter	Level tank measurement	cm
FT-200	Flow sensor/transmitter	Output flow measurement	%

[6] Faults have very low prior probabilities.

[7] See [Poole *et al.*, 1998] for a nice overview of Bayesian inference.

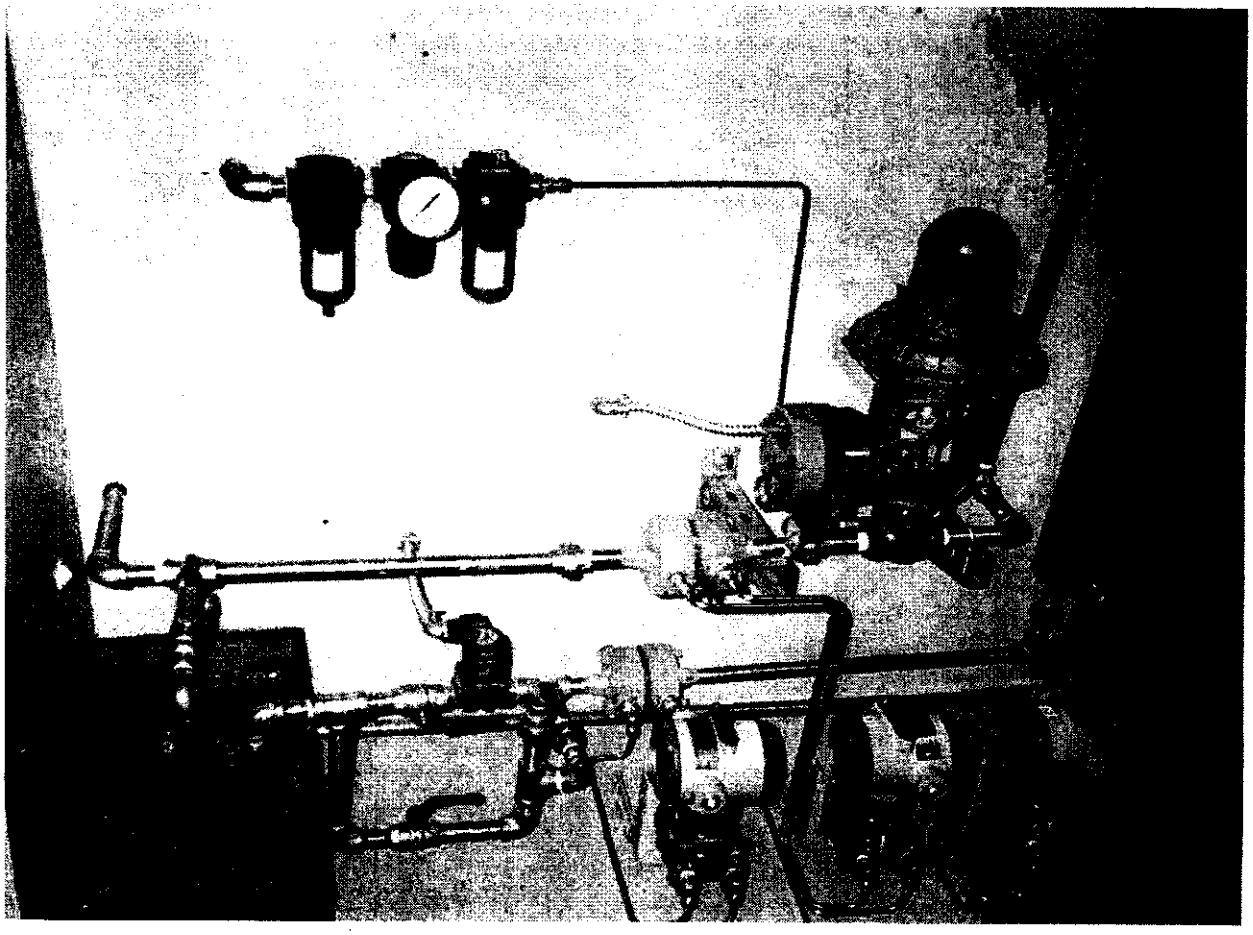

Figure 5.8: Level Tank Process. The instrumentation shown has industrial characteristics. We are using orifice plates as sensors, and pressure differential transmitters.

5.3.2 Modelling

4 Discrete modes.

We engineered four discrete operational modes under conditions shown in Table 5.7. Manual valves (V_1 and V_2) satisfy $0 \leq O_1 \leq O_2 \leq 100\%$ and $0 \leq O_3 \leq 100\%$. O_1, O_2 and O_3 were adjusted to work in linear regimes.

Following the same mathematical methods (described for the industrial dryer), we get the *JMLG* parameters, Table 5.8.

5 Discrete modes.

To increase the complexity a little, we added another discrete operating state, shown in Table 5.9. The *JMLG* parameters for the new system are given in Table 5.10. Ideally, the parameters for the first four states ($z_t = 1, \ldots, 4$) would have been the same for both systems; however, due to difficulties in maintaining certain conditions we could only reproduce approximate values. This is an interesting point to discuss later, but for the time being, this variability in the process allows us to test the robustness of the Particle Filtering algorithms.

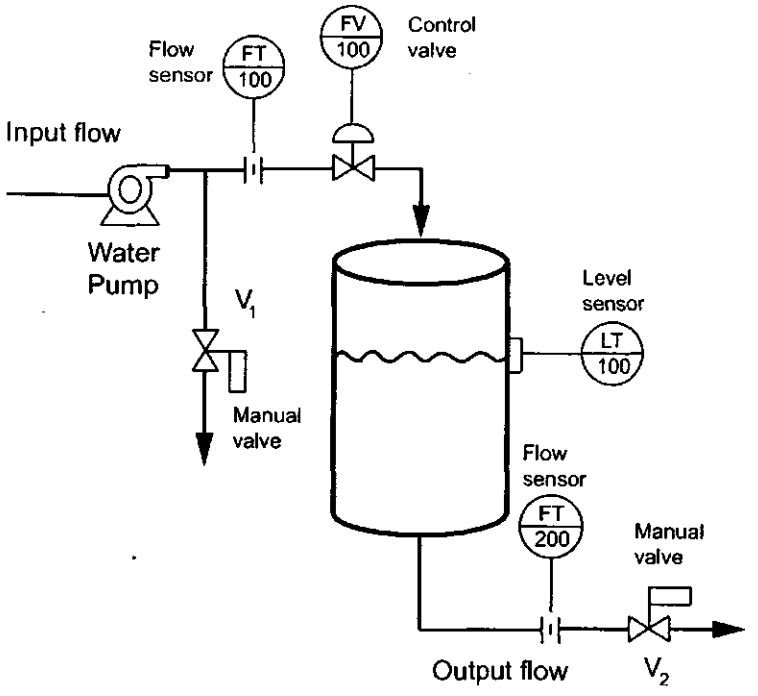

Figure 5.9: Level Tank. Instrumentation diagram. Using only the level sensor/transmitter (LT-100), we want to diagnose the different discrete modes.

Table 5.7: Level tank (4 discrete modes). Operating Conditions.

z_t	Model name	V_1	FV-100	V_2
1	Low In-Flow	100 %	40 %	100 %
2	Medium In-Flow	O_2 % (partly open)	40 %	100 %
3	High In-Flow	O_1 % (almost closed)	40 %	100 %
4	Low Out-Flow	100 %	40 %	O_3 % (partly open)

Table 5.8: Level tank (4 discrete modes). *JMLG* parameters.

z_t	$x_0(z_t)$	$A(z_t)$	$B(z_t)$	$C(z_t)$	$D(z_t)$	$F(z_t)$	$G(z_t)$
1	15.54	0.9801	0.008	1	0.08	0.3099	0
2	31.40	0.9829	0.008	1	0.08	0.5368	0
3	68.84	0.9910	0.008	1	0.08	0.6163	0
4	27.61	0.9872	0.008	1	0.08	0.3544	0

Table 5.9: Level tank (5 discrete modes). Operating Conditions.

z_t	Model name	V_1	FV-100	V_2
1	Low In-Flow	100 %	40 %	100 %
2	Medium In-Flow	O_2 % (partly open)	40 %	100 %
3	High In-Flow	O_1 % (almost closed)	40 %	100 %
4	Low-In/Low-Out	100 %	40 %	O_3 % (partly open)
5	Medium-In/Low-Out	O_2 % (partly open)	40 %	O_3 % (partly open)

Table 5.10: Level tank (5 discrete modes). JMLG parameters.

z_t	$x_0(z_t)$	$A(z_t)$	$B(z_t)$	$C(z_t)$	$D(z_t)$	$F(z_t)$	$G(z_t)$
1	15.93	0.9864	0.05	1.0	0.05	0.2169	0
2	30.61	0.9886	0.05	1.0	0.05	0.3484	0
3	69.00	0.9880	0.05	1.0	0.05	0.8285	0
4	24.39	0.9894	0.05	1.0	0.05	0.2574	0
5	41.52	0.9925	0.05	1.0	0.05	0.3114	0

5.3.3 Diagnosis/estimation tests

Several random sequences were implemented in order to obtain real data for both groups of discrete states.

4 Discrete modes.

More than 20 random sequences were implemented using the following transition matrix and initial prior probabilities, equation (5.5). A representative sequence and modelling results are given in Figure 5.10.

$$P(z_t|z_{t-1}) = \begin{bmatrix} 0.996 & 0.002 & 0.00 & 0.002 \\ 0.002 & 0.996 & 0.002 & 0 \\ 0 & 0.004 & 0.996 & 0 \\ 0.002 & 0 & 0 & 0.998 \end{bmatrix} \quad P(z_0) = \begin{bmatrix} 0.99 \\ 0.001 \\ 0.001 \\ 0.008 \end{bmatrix}' \tag{5.5}$$

5 Discrete modes.

More than 15 random sequences were implemented using the following transition matrix and initial probabilities, equation (5.6). Figure 5.11 shows a representative random sequence and modelling results.

$$P(z_t|z_{t-1}) = \begin{bmatrix} 0.995 & 0.002 & 0.002 & 0.001 & 0.0 \\ 0.001 & 0.997 & 0.001 & 0.0 & 0.001 \\ 0.001 & 0.002 & 0.997 & 0.0 & 0.0 \\ 0.001 & 0.0 & 0.0 & 0.998 & 0.001 \\ 0.0 & 0.001 & 0.0 & 0.001 & 0.998 \end{bmatrix} \quad P(z_0) = \begin{bmatrix} 0.996 \\ 0.001 \\ 0.001 \\ 0.001 \\ 0.001 \end{bmatrix}' \tag{5.6}$$

Other random sequences for the level tank system are included in Appendix D section D.2.1.

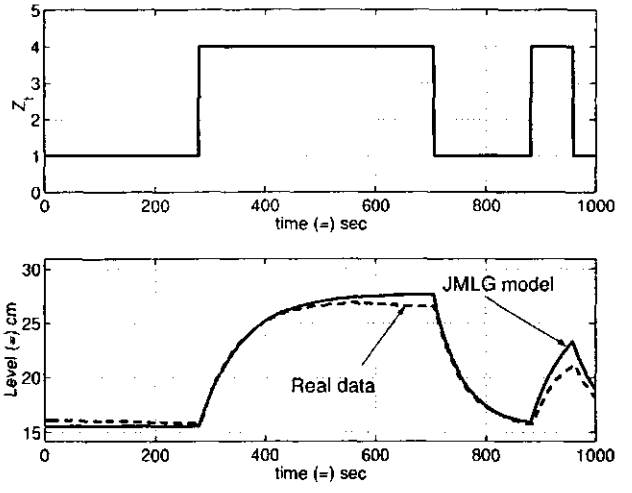

Figure 5.10: Level tank (4 discrete modes). Random sequence No. 1. The upper graph shows the discrete modes over time, while the lower graph compares the real level with the synthetic data obtained using the *JMLG* model for the same discrete modes.

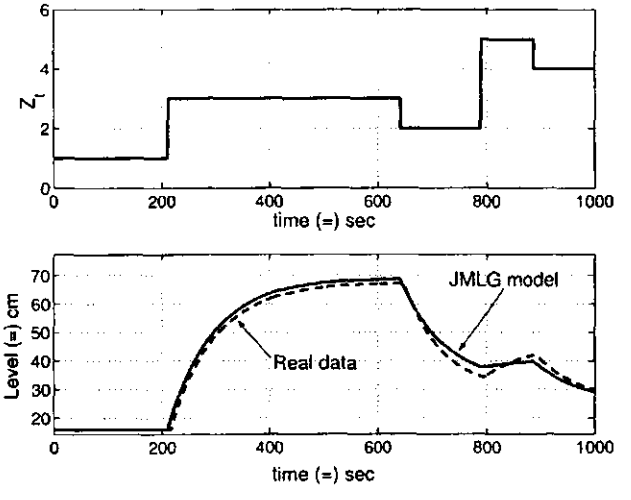

Figure 5.11: Level tank (5 discrete modes). Random sequence No. 1.

5.3.4 Results

4 Discrete modes.

Figure 5.12 depicts diagnosis error versus number of particles, and diagnosis error versus computing time per time-step, respectively, for random sequence No. 1. Numerical data for these figures and other results for this domain are included in Appendix D section D.2.2. It is important to note that we used a differential pressure sensor-transmitter in the level tank. This kind of instrument usually has high noise levels.

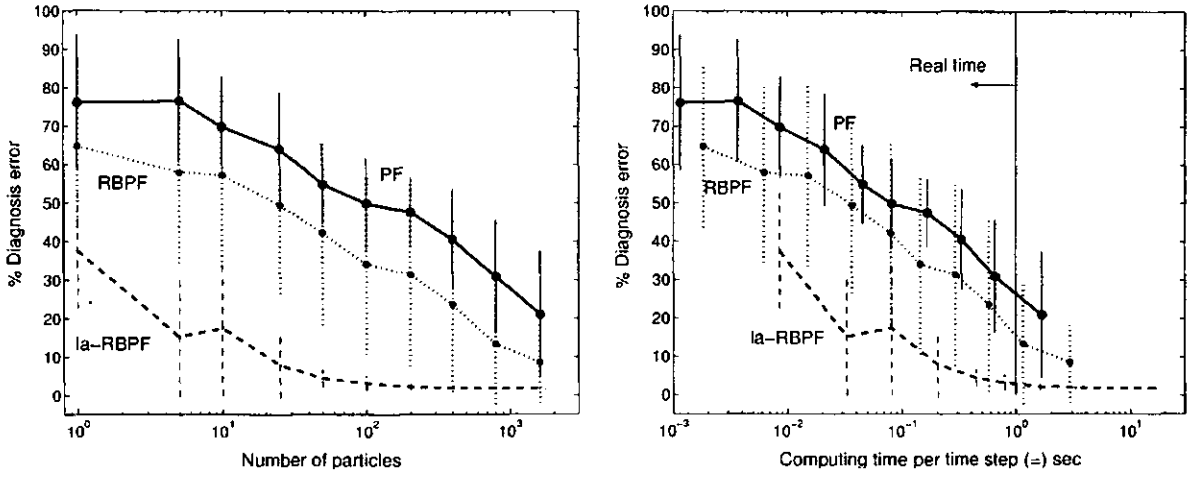

Figure 5.12: Level tank (4 discrete modes). Diagnosis error for random sequence No. 1. The left graph shows diagnosis error versus number of particles, while the right graph shows diagnosis error versus computing time per time step.

To illustrate the variability of the diagnosis error, we show the same results as a box and whisker plot in Figure 5.13. The boxes have lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of a box to show the extent of the rest of the data. The boxes are notched. Notches represent a robust estimate of the uncertainty about the medians for box-to-box comparison. Outliers are data values beyond the ends of the whiskers; outliers are shown as circles. Note how the *la-RBPF* quartiles are closer than the Particle Filtering and Rao-Blackwellized Particle Filtering quartiles.

5 Discrete modes.

Figure 5.14 shows diagnosis error versus number of particles and diagnosis error versus computing time per time step, respectively, for random sequence No. 1.

Other results are included in Appendix D section D.2.2.

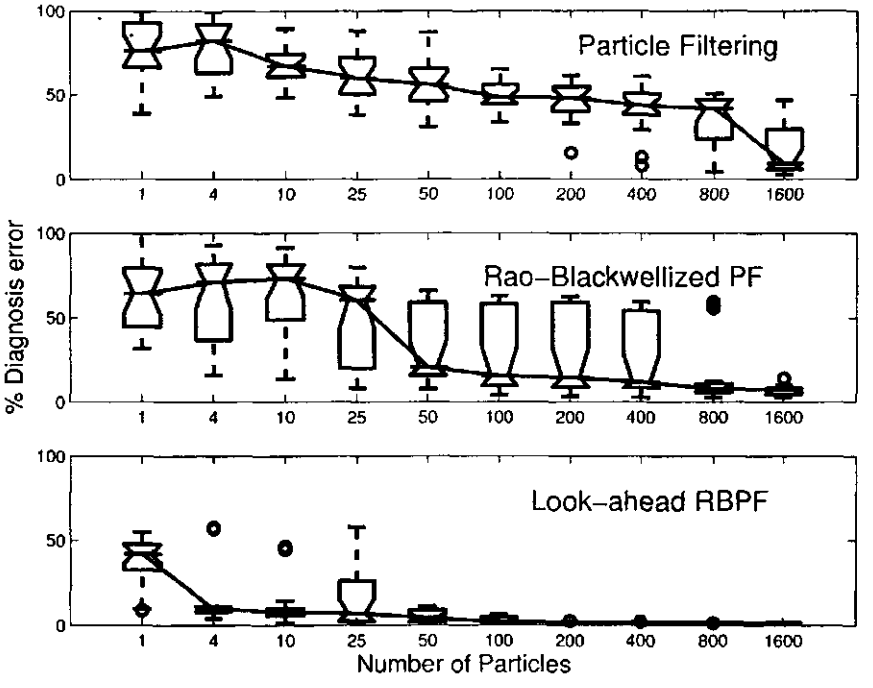

Figure 5.13: Box and whisker plots for each Particle Filtering algorithm. *la-RBPF* always works significantly better than regular *PF* and *RBPF* for each number of particles.

5.4 Industrial heat exchanger

5.4.1 Description

We used an industrial heat exchanger in a horizontal arrangement, model BEU-660, with 11 Cu-Ni (90-10) 3/4 inch external diameter tubes, BWG-20 triangular arrangement, Figure 5.15. This exchanger heats 10 gpm of water in the range of $25°C$ to $70°C$ with steam at 5 Kg/cm^2. The container is 6 inches in external diameter and 70 inches in length. The equipment is operated automatically by a distributed industrial control system, the Honeywell TDC 3000. Figure 5.16 is a conceptual diagram; the main instrumentation is described in Table 5.11.

These instruments have analog (4-20 mA) communication with a Honeywell TDC 3000 Distributed Control System, which in turn has digital communication with a computer through the PCIM (Personal Computer Interface Module) [Morales-Menéndez, 1992]. This was our data acquisition system. Additionally, we configured two PID controllers (FIC-202 and FIC-203) to control the input water flow and the steam flow during the tests. There are many additional accessories and devices, but we omit them from the instrumentation diagram for the sake of simplicity.

5.4.2 Modelling

The heat exchanger is a nonlinear process. Based on the input water flow, fixed by the FIC-203 controller, we found five different operating regimes, Table 5.12. The *JMLG* parameters for these regimes are shown in Table 5.13. Some

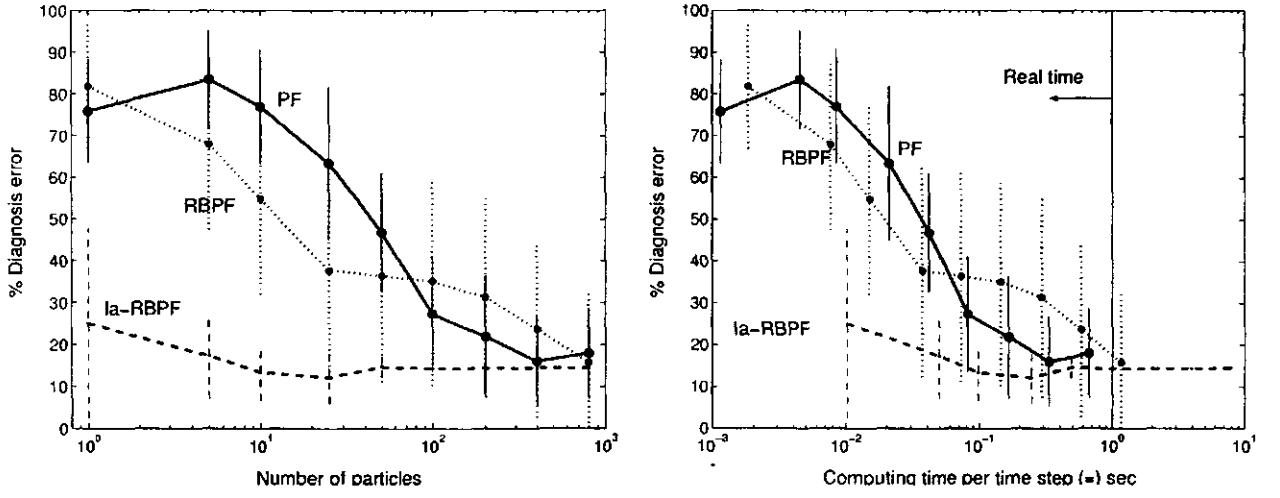

Figure 5.14: Level tank (5 discrete modes). Diagnosis error for random sequence No. 1. The left graph shows diagnosis error versus number of particles, while the right graph shows diagnosis error versus computing time per time step.

Table 5.11: Heat exchanger. Instrumentation.

Tag-name	Functional name	Description	Units
FT-203	Flow sensor/transmitter	Input water flow measurement	%
TT-203	Temperature sensor/transmitter	Input water temperature	^oC
FV-203	Flow valve	Input water control valve	%
FIC-203	Flow indicator controller	Water flow PID controller	
FT-202	Flow sensor/transmitter	Steam flow measurement	%
FV-202	Flow valve	Steam flow control valve	%
FIC-202	Flow indicator controller	Steam flow PID controller	
TT-201	Temperature sensor/transmitter	Output water temperature measurement	^oC

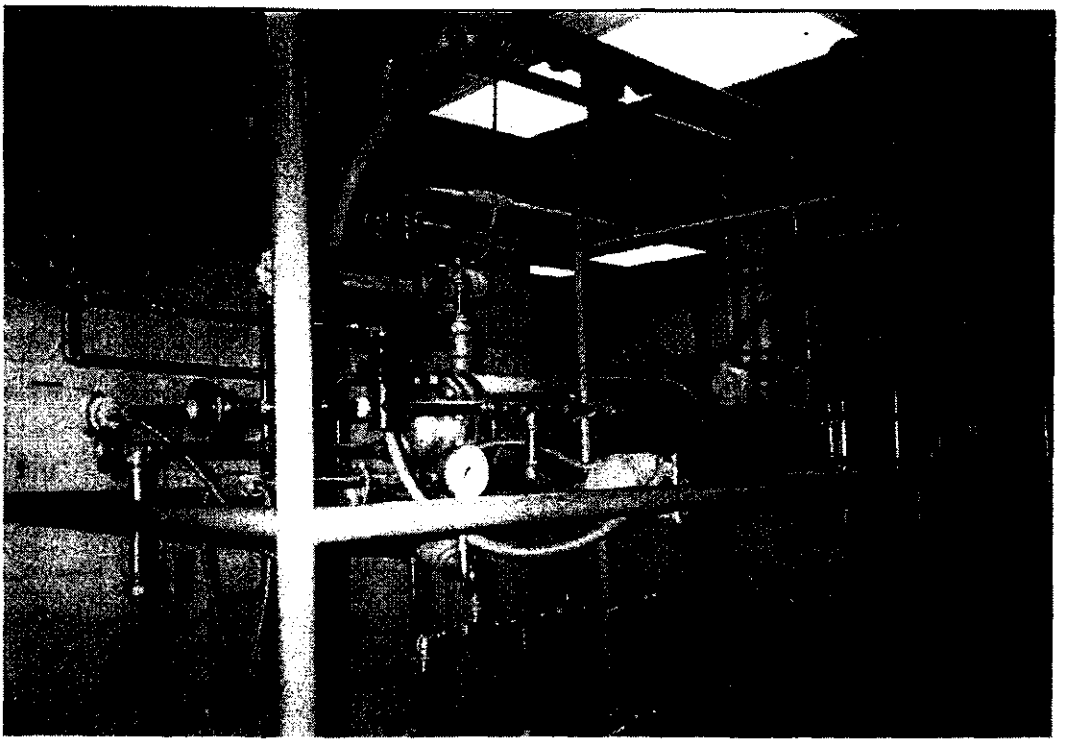

Figure 5.15: Industrial heat exchanger. This exchanger heats 10 gpm of water in the range of $25°C$ to $70°C$ with steam at 5 Kg/cm^2.

intermediate results from the *Expectation-Maximization*[8] method are shown in Appendix D section D.3.1.

Table 5.12: Heat exchanger. Operating conditions.

z_t	Model name	Input flow	Steam flow	Water temperature
1	Very high flow	65 %	32 %	39.69 oC
2	High flow	57 %	32 %	41.68 oC
3	Normal flow	49 %	32 %	44.05 oC
4	Low flow	41 %	32 %	47.26 oC
5	Very low flow	33 %	32 %	51.40 oC

5.4.3 Diagnosis/estimation tests

We implemented 20 random sequences using the transition matrix and initial probabilities given in equation (5.7). Figure 5.17 shows a representative random sequence, called random sequence No. 1 for this domain. The upper graph represents the discrete modes implemented, and the lower graph shows how the real output water temperature

[8] We thank Zoubin Ghaharamani (University College London) for his linear dynamic system *EM* code and his suggestions.

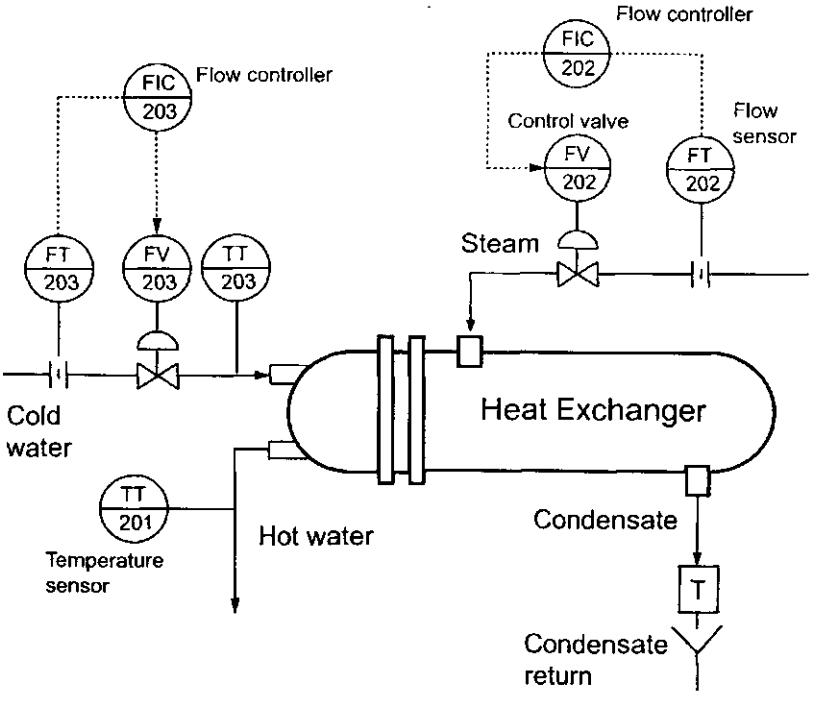

Figure 5.16: Industrial heat exchanger. Instrumentation diagram. Two PID controllers (FIC-203 and FIC-202) were configured to control the input water flow and the steam flow during experiments.

changed over time. The lower graph also shows synthetic data which was generated for comparison. Another similar random sequence is included in Appendix D section D.3.2.

$$P(z_t|z_{t-1}) = \begin{bmatrix} 0.997 & 0.0015 & 0.001 & 0.0005 & 0.0 \\ 0.001 & 0.997 & 0.001 & 0.00075 & 0.00025 \\ 0.0005 & 0.002 & 0.997 & 0.001 & 0.0 \\ 0.00025 & 0.00075 & 0.001 & 0.997 & 0.001 \\ 0.0 & 0.0005 & 0.001 & 0.0015 & 0.997 \end{bmatrix} \quad P(z_0) = \begin{bmatrix} 0.0005 \\ 0.001 \\ 0.997 \\ 0.001 \\ 0.0005 \end{bmatrix}' \tag{5.7}$$

Table 5.13: Heat exchanger. *JMLG* parameters.

z_t	$x_0(z_t)$	$A(z_t)$	$B(z_t)$	$C(z_t)$	$D(z_t)$	$F(z_t)$	$G(z_t)$
1	39.63	0.9818	0.005	1.0	0.05	0.7209	0
2	41.69	0.9817	0.005	1.0	0.05	0.7591	0
3	44.03	0.981	0.005	1.0	0.05	0.7970	0
2	47.23	0.9858	0.005	1.0	0.05	0.6695	0
2	51.44	0.9786	0.005	1.0	0.05	1.0999	0

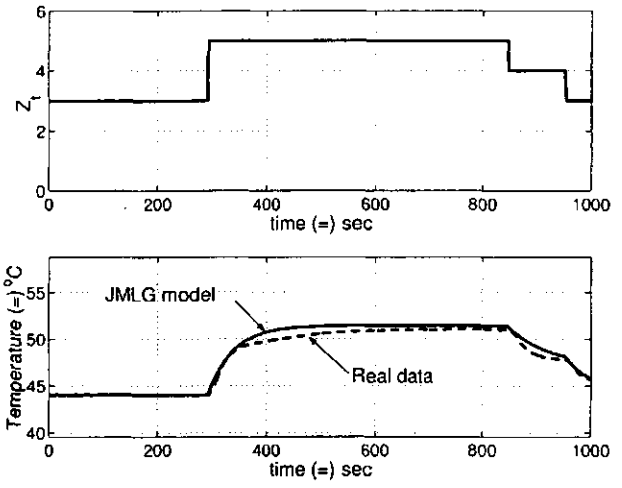

Figure 5.17: Industrial heat exchanger. Random sequence No. 1. The upper graph shows the discrete modes over time, while the lower graph compares the real level with the synthetic data obtained using the *JMLG* model for the same discrete modes.

5.4.4 Results

Figure 5.18 shows the diagnosis error for random sequence No. 1. The left graph shows diagnosis error versus number of particles for the three Particle Filtering algorithms; the right graph shows diagnosis error versus computing time per time step. Other results for this domain are included in Appendix D section D.3.3.

Box and whisker plots (Figure 5.19) are another interesting way to look at the results. Note how the *la-RBPF* quartiles are closer than those of the Particle Filtering and Rao-Blackwellized Particle Filtering algorithms.

A more complete way to show the Particle Filtering results is to plot the probability distribution $p(z_t|y_{1:t})$ over time. Figures 5.20 show the probability distribution $p(z_t|y_{1:t})$ every 25th time step. The upper graph shows the results for Particle Filtering, while the lower graph shows Rao-Blackwellized Particle Filtering. Both used $N = 5$ particles.

Figures 5.21 show the results for *la-RBPF*. The upper graph shows the results using the same $N = 5$ particles, while the lower graph was built using 400 particles.

Variation in the transition matrix

A very simple robustness test was implemented for the three Particle Filtering algorithms. Some variations were realized in the probabilities of

- Prior distribution of the discrete modes $p(z_0)$
- The less representative discrete mode ($z_t = 1$) in the transition matrix
- The most representative discrete mode ($z_t = 3$) in the transition matrix

However, as we expect there is no change in the diagnosis error, see Table D.7 in Appendix D, section D.3.3 where the probabilities of the most representative discrete mode were modified.

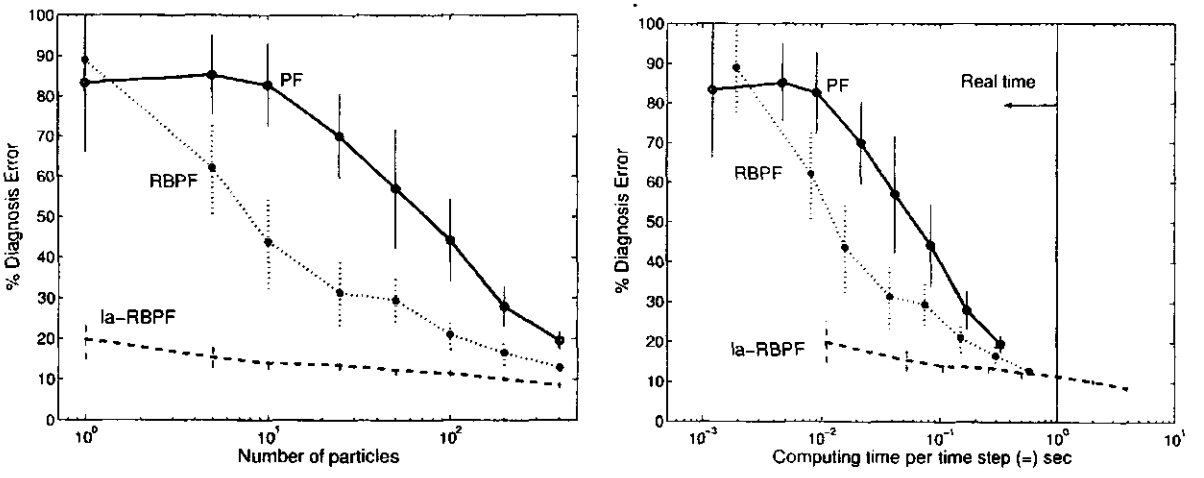

Figure 5.18: Heat exchanger. Diagnosis error for random sequence No. 1. The left graph shows diagnosis error versus number of particles; the right graph shows diagnosis error versus computing time per time step.

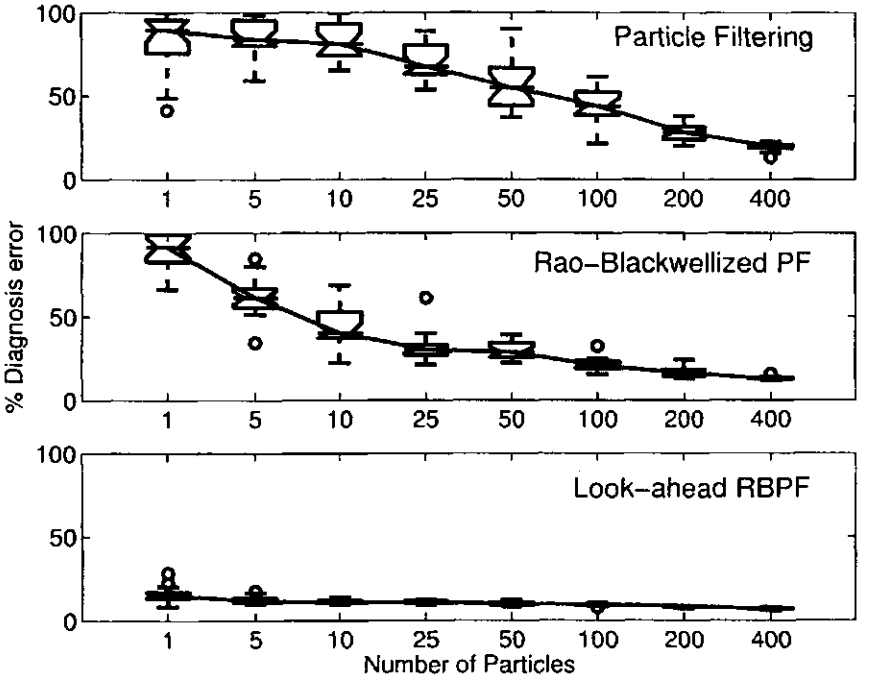

Figure 5.19: Box and whisker plots for each Particle Filtering algorithm. *la-RBPF* always works significantly better than regular *PF* and *RBPF* for each number of particles.

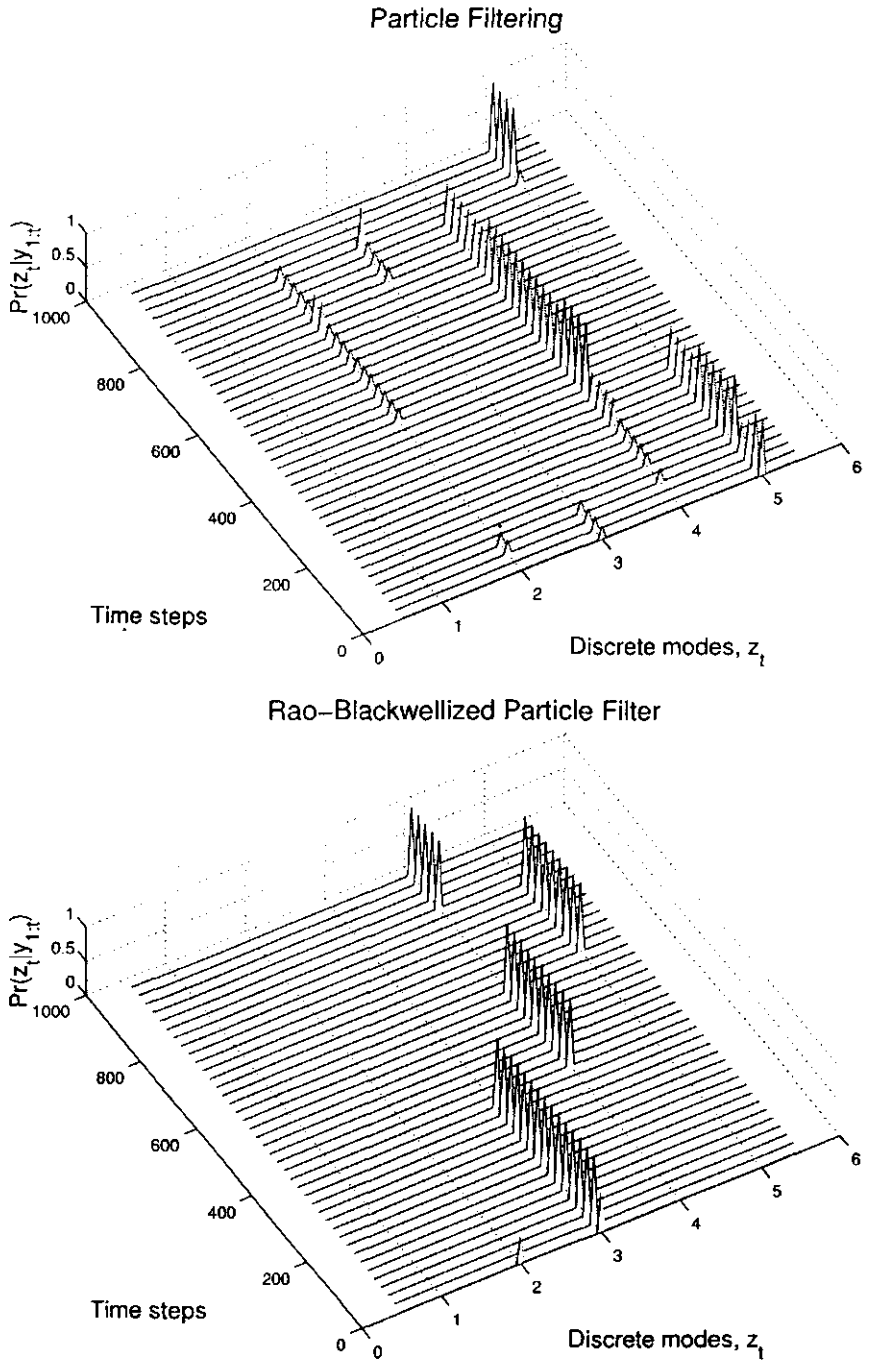

Figure 5.20: Probability distribution $p(z_t|y_{1:t})$ over time. The upper graph shows $p(z_t|y_{1:t})$ as approximated by the standard Particle Filtering algorithm, while the lower graph uses Rao-Blackwellized Particle Filtering. Both approximations used five particles.

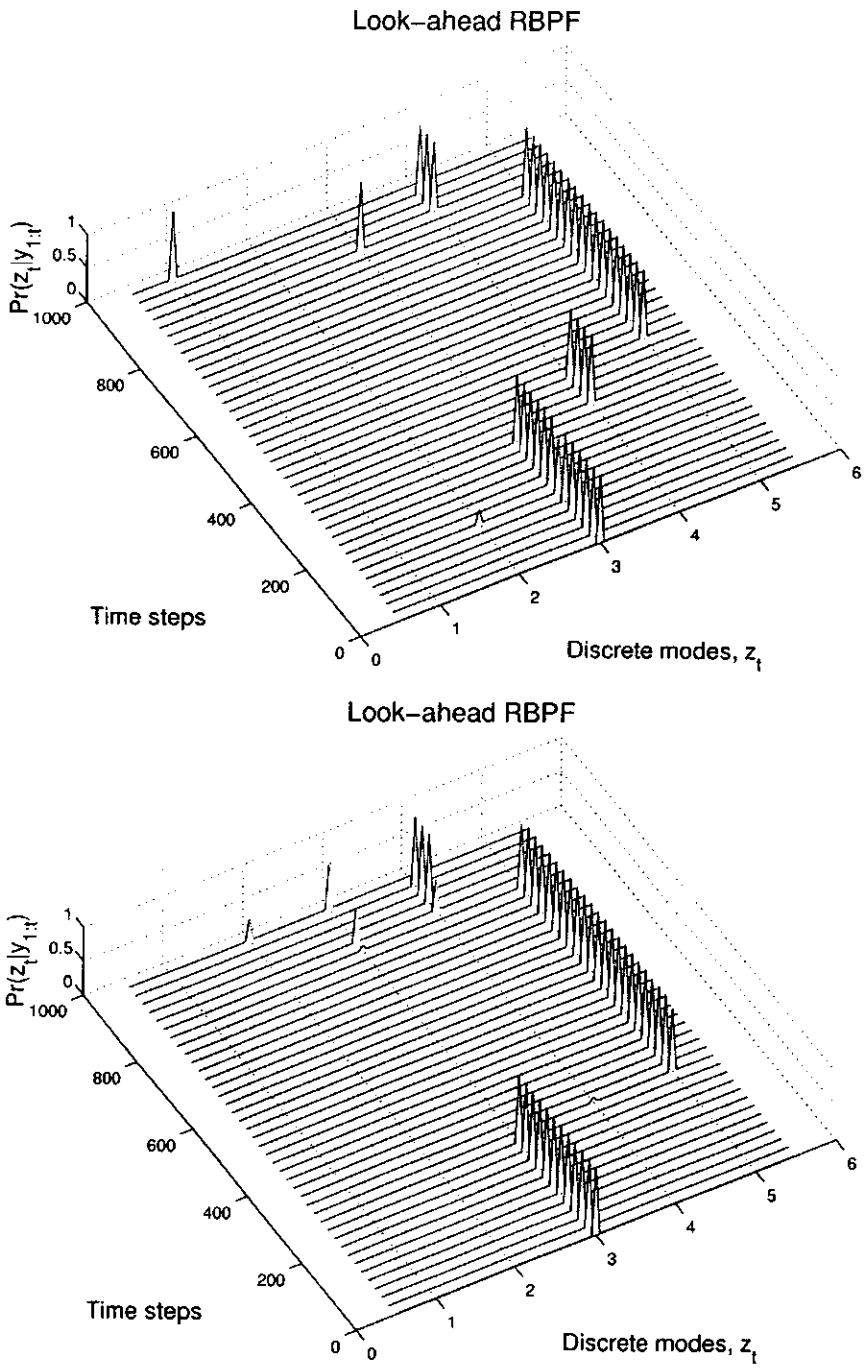

Figure 5.21: Probability distribution $p(z_t|y_{1:t})$ over time. Both graphs were generated with the *look-ahead Rao-Blackwellized Particle Filtering* algorithm. For the upper graph we used 5 particles; the lower graph was built using 400 particles.

5.5 Mobile robot

The previous industrial processes have normal and faulty states, while robots must further deal with changing environmental states.

5.5.1 Description

José[9], Figure 5.22, is a Real World Interfaces B-14 mobile robot with a B-12 base. José resides at the Laboratory for Computational Intelligence[10] at the University of British Columbia, Canada.

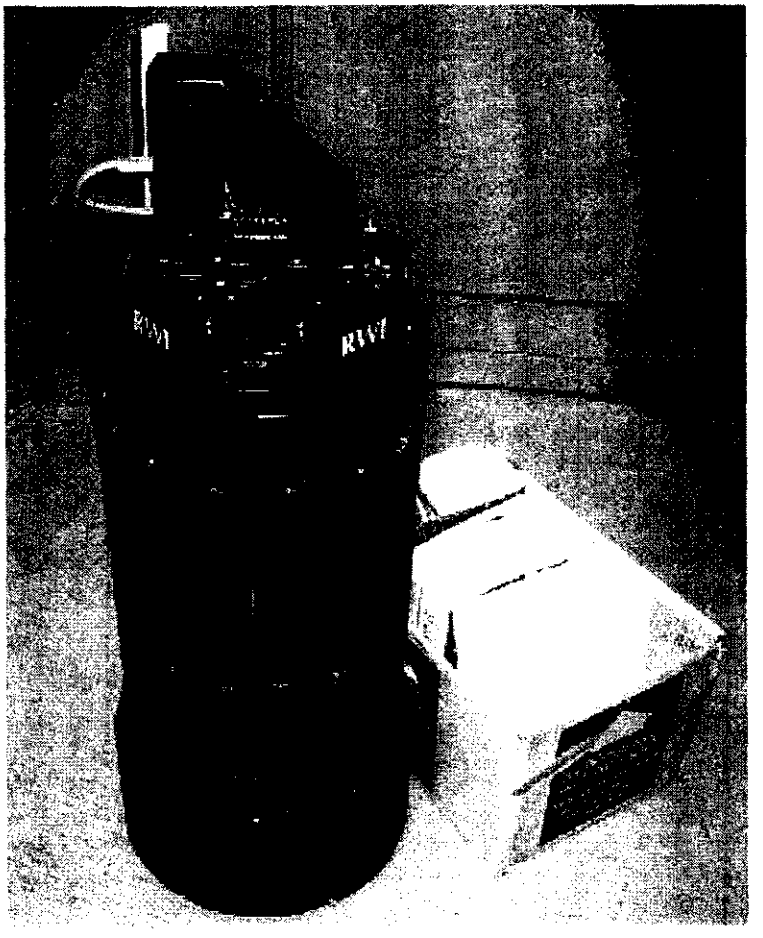

Figure 5.22: José, the mobile robot. José, winner of the 2001 AAAI Hors d'Oeuvres competition [Elinas *et al.*, 2002], is a Real World Interfaces B-14 mobile robot with a B-12 base.

The base unit contains two separate motors, one for translation and one for rotation. The translational motor drives all three wheels, resulting in excellent traction. The rotational motor turns all three wheels in unison, so they are always pointing the same way. Thus there is no concept of *front* or *back* wheels - a given wheel can end up in

[9]He's named after José Narváez, the first European to explore Georgia Strait, BC, Canada.

[10]We thank Don Murray (University of British Columbia) for getting us started with José.

the front (relative to the direction of travel), at the back, or anywhere in between. José can move along curved paths by simultaneously translating and rotating.

The B-12's motors are stepping motors, controlled by a varying pulse width. Actual translation and rotational speed are measured by optical shaft encoders - one for translation and one for rotation. Individual wheels do not have separate encoders, so are assumed to all be translating or rotating at the same speed. There are commands to set the desired speed and acceleration. A feedback control loop monitors the actual values and adjusts the motor pulse width to achieve the desired values. Simpler open-loop commands are also available - these bypass the control loop and send a fixed pulse width directly to the motors. José also has a stereo camera and bump, infrared, and sonar sensors.

Software

José's on-board computer runs Linux, BaseServer base-unit control software, Triclops depth-mapping software, and a number of higher-level custom modules for localization, path planning, etc. A shared memory architecture allows the simultaneous running of multiple high-level modules, Figure 5.23.

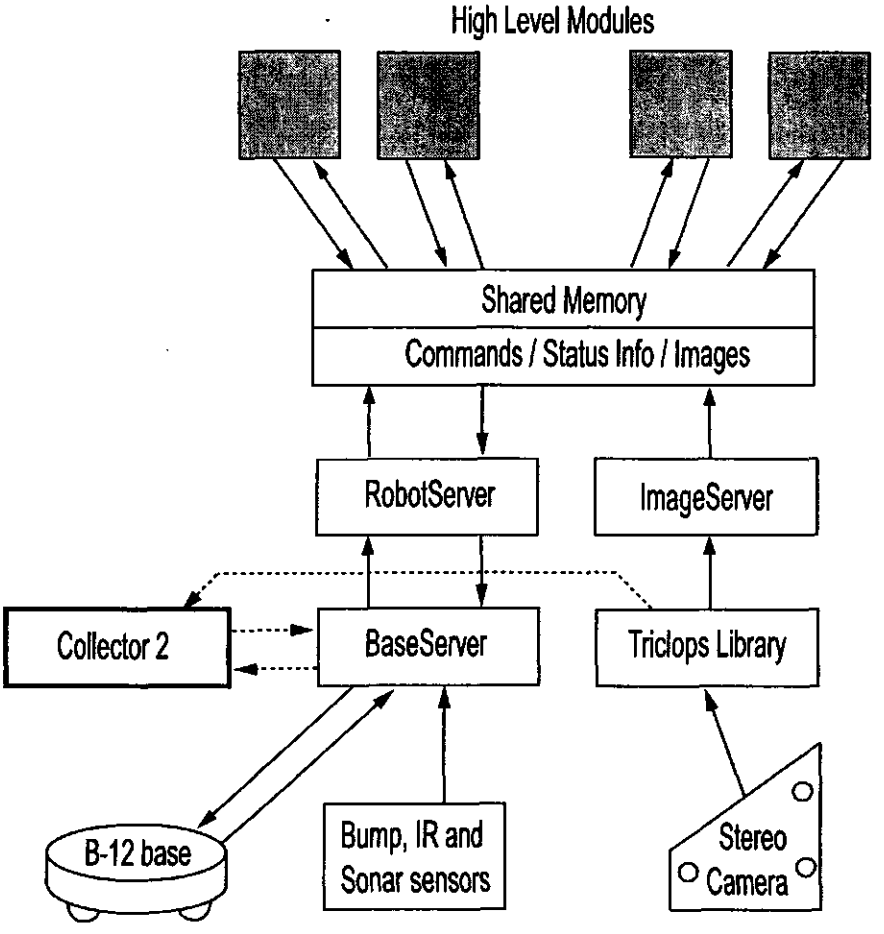

Figure 5.23: José's software architecture.

An on-line monitoring system generally requires access to low-level information and commands (such as mo-

tor currents and shaft velocities) that other modules do not care about. This functionality was not available in the RobotServer/shared memory architecture, so for the purposes of our experiments a stand-alone program was written[11]. *Collector2* talks directly to BaseServer and the Triclops library, allowing us to queue any commands, set any sampling rate, and get any available data.

5.5.2 Modelling

We chose to monitor states involving movement, specifically, states in which there is an extra load, see Table 5.14. Normal operation conditions corresponds to a null extra load. A change to another state can occur if José bumps into or snags something. To simulate these states repeatably, we manually introduced the loads.

We implemented these conditions for movement across a *smooth* floor, Table 5.14, and also for a *tiled* floor. José's speed was the observable variable y_t; we monitored this variable every 0.1 sec.

Table 5.14: José. Operating conditions.

z_t	Model name	Extra load	Floor
1	Normal	0	Smooth
2	Low load	0.8 kg	Smooth
3	Medium load	1.6 kg	Smooth
4	High load	2.4 kg	Smooth

Table 5.15 shows the *JMLG* parameters for the smooth floor. Parameters were also obtained for the tiled floor.

Table 5.15: José on the smooth floor. JMLG parameters.

z_t	$x_0(z_t)$	$A(z_t)$	$B(z_t)$	$C(z_t)$	$D(z_t)$	$F(z_t)$	$G(z_t)$
1	0.56187	0.72977	0.005	1.0	0.005	0.15183	0
2	0.5037	0.7385	0.005	1.0	0.005	0.13171	0
3	0.4265	0.69264	0.005	1.0	0.005	0.13108	0
4	0.35805	0.74912	0.005	1.0	0.005	0.08982	0

5.5.3 Diagnosis/estimation tests

We engineered a transition matrix $P(z_t|z_{t-1})$ and initial state $z_0 \sim P(z_0)$, equation (5.8).

$$P(z_t|z_{t-1}) = \begin{bmatrix} 0.99425 & 0.005 & 0.0005 & 0.00025 \\ 0.005 & 0.9895 & 0.005 & 0.0005 \\ 0.0005 & 0.005 & 0.9895 & 0.005 \\ 0.00025 & 0.0005 & 0.005 & 0.99425 \end{bmatrix} \quad P(z_0) = \begin{bmatrix} 0.9825 \\ 0.01 \\ 0.005 \\ 0.0025 \end{bmatrix}' \tag{5.8}$$

[11] We thank Jim Mutch (University of British Columbia) for developing this interface, and for his invaluable support during the experimental tests.

More than 30 random sequences were implemented on the smooth and tiled floors, in which we physically changed the robot's operating conditions (load) at random times. Figure 5.24 shows two representative sequences for both floors. We can see that the *JMLG* model successfully describes the behaviour on both surfaces. The upper graphs show the discrete mode of operation over time, and the lower graphs show the real and simulated data (*JMLG* model). The left graphs correspond to the smooth floor and the right graphs to the tiled floor. Note the noisy signal for the bumpy, tiled floor.

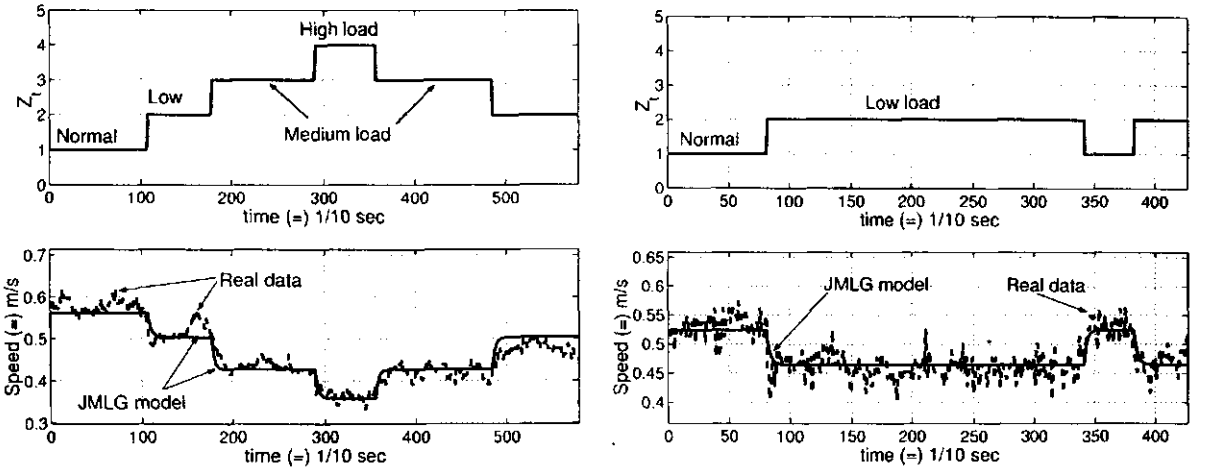

Figure 5.24: José. Random sequences for smooth and tiled floor. The upper graphs show the discrete mode over time, while the lower graphs show the real speed and synthetic data (*JMLG* model) for these discrete modes. The left graphs correspond to the smooth floor, and the right graphs to the tiled floor. (Note the noisy signal for the tiled floor.)

5.5.4 Results

As for previous domains, we show diagnosis error versus number of particles, and diagnosis error versus computing time per time step, Figure D.22. Both graphs are for the smooth floor. Additional results for the mobile robot are included in Appendix D section D.4.1.

5.5.5 Summary

The proposed learning procedure for the *jump Markov linear Gaussian* parameters was tested with several real data sets taken from four real-world systems. From the modelling point of view, the results had acceptable residuals; however, for the inference task (our main goal), the results were excellent.

Intermediate results indicate that the proposed learning procedure, which is iterative, has several time consuming steps. There is an opportunity to improve the overall procedure by automating or integrating some steps. Implementing recursive versions of the *Least Squares Estimation* and *Expectation-Maximization* algorithms would provide great advantages, especially during model maintenance, in which we must train, learn and test the *JMLG* models continuously. Recursive versions would also allow us to adapt and capture the changing conditions of industrial processes or the changing environments of autonomous systems such as mobile robots.

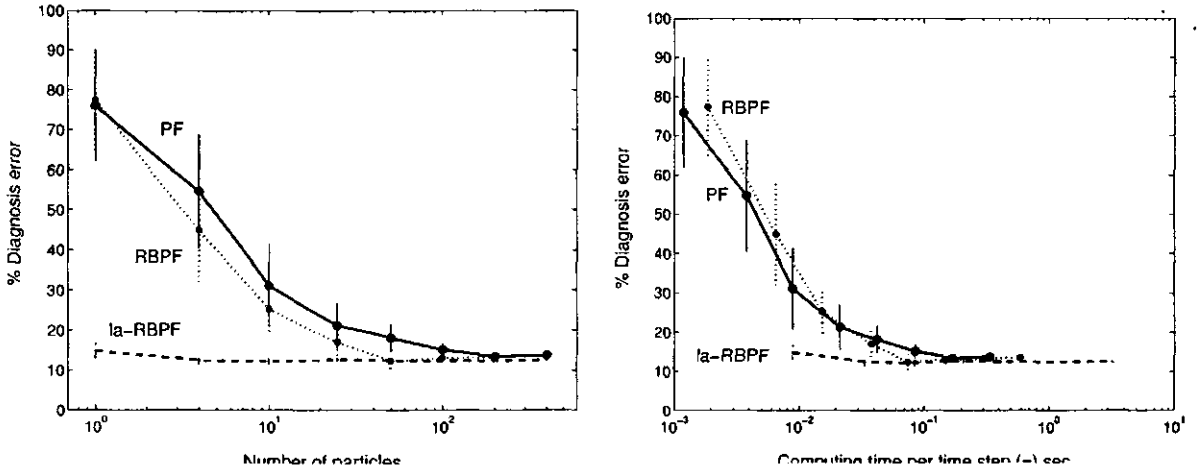

Figure 5.25: José. Diagnosis error on smooth floor. The left graph shows diagnosis error versus number of particles; the right graph shows diagnosis error versus computing time per time step. For this domain, 0.1 sec is the maximum computing time allowed per time step for real time diagnosis.

The three Particle Filtering inference algorithms were tested and compared in terms of diagnosis error versus number of particles and computing time per time step. *look-ahead Rao-Blackwellized Particle Filtering* significantly outperformed *PF* and RBPF in the four real-world domains. Generally speaking, *la-RBPF* gives lower diagnosis error and lower variance, both per number of particles and computing time per time step.

The implementation of faulty conditions is difficult and highly risky in real-world domains. In the following Chapter 6, we use simulation to test our inference algorithms in more complex conditions.

Chapter 6

Simulated systems

6.1 Introduction

In order to test our inference algorithms on specific aspects not shown in the experimental implementations, we simulated two systems, a Continuous Stirred Tank Reactor (*CSTR*) and the previous mobile robot system. Basically, we increased:

- the number of discrete modes
- the number of continuous state space variables
- the noise level in process and measurement signals

The CSTR system is a highly nonlinear process. It has several interesting features and has been used by other researchers [Chen and Howell, 2001; Dash *et al.*, 2003] to test fault diagnosis.

As in Chapter 5, each domain is presented in four sections: *description*, *modelling*, *diagnosis/estimation tests*, and *results*.

Finally a summary is included at the end of this chapter.

6.2 Continuous Stirred Tank Reactor

6.2.1 Description

In the jacketed chemical reactor (*CSTR*) shown in Figure 6.1, a second-order exothermic reaction ($2A \rightarrow B$) takes place, in which 2 components A react irreversibly and at specific reaction rate k to form a product, B. The reaction rate constant k follows the Arrhenius equation (6.1). According to this equation, the effect of temperature, $T_r(t)$, on the specific reaction rate k is usually exponential. This exponential temperature dependence represents one of the most severe nonlinearities in chemical engineering systems. The overall reaction rate R is defined as the rate of change of moles of any component per volume due to chemical reaction divided by that component's stoichiometric coefficient. Because of this reaction, we have $R = kC_A^2$. Then, the overall rate R will vary with temperature, $T_r(t)$, and with the concentration of the reactant C_A raised to the 2nd power (second-order reaction). As we can see, this

term R is highly nonlinear.

$$k(T_r(t)) = k_0 e^{\left[\frac{-a}{T_r(t)+460}\right]} \tag{6.1}$$

6.2.2 Modelling

The mathematical model for this *CSTR* involves a *mass balance on A*, in which the flow of moles of component A into the system, minus the flow of moles of A out of the system, plus the rate of formation of moles of A component form chemical reactions is equal to time rate of change of moles of A component inside system. This concept is expressed by equation (6.2), commonly known as a *component continuity equation* [Smith, 1972].

The first law of thermodynamics puts forward the principle of conservation of energy. The mathematical model must include an *enthalpy balance on reacting mass*, and an *enthalpy balance on jacket* (water is flowing through the jacket). In this case, the flow of internal energy into the system, minus the flow of internal energy out of the system, plus the heat added to the system by reaction is equal to the rate of change of internal energy inside the system. The balance on reacting mass is given by equation (6.3), and the balance on the jacket by equation (6.4).

$$V\rho\frac{dC_A}{dt} = W(C_{Ai} - C_A) - kC_A^2 V\rho \tag{6.2}$$

$$V\rho C_P\frac{dT_r}{dt} = WC_p(T_{ri} - T_r) - UA(T_r - T_{jo}) + (-\Delta H)VkC_A^2 \tag{6.3}$$

$$M_j C_{pj}\frac{dT_{jo}}{dt} = UA(T_r - T_{jo}) - W_j C_{pj}(T_{jo} - T_{ji}) \tag{6.4}$$

Some assumptions were made to derive these equations; see [Smith, 1972] and [Luyben, 1989] for a full explanation. These equations represent a very simplified nonlinear *CSTR* model (the t functionality was omitted for clarity). Nevertheless, this simpler model captures the nonlinearity we are interested in [Luyben, 1989]. We only measure the following 3 variables: the output concentration $C_A(t)$, the reactor temperature $T_r(t)$, and the output jacket temperature $T_{jo}(t)$. See Appendix E, Table E.1 for a complete definition of the variables; Figure 6.1 conveys their meaning graphically.

Figure 6.1 also shows some instruments for monitoring and control purposes. Table 6.1 gives a complete description of this instrumentation.

The nonlinear model described by equations (6.2-6.4) was linearized to build the *JMLG* model (Appendix E section E.1.2). Then two group of discrete modes were tested:

- *Group 1*: 4 discrete modes
- *Group 2*: 10 discrete modes

4 discrete modes.

We consider a fouled surface (dirty surface) in the jacket as a possible faulty point (of course, there are many possible faulty points [Chen and Howell, 2001] in this system). A fouled surface can be caused by normal operating conditions over an extended time, or by stochastic problems such as cooling water with a high concentration of minerals or salts. Surface fouling reduces the global heat transfer coefficient $U(t)$ in the mathematical model. We defined four of the possible discrete modes for this nonlinear multi-variable system, Table 6.2.

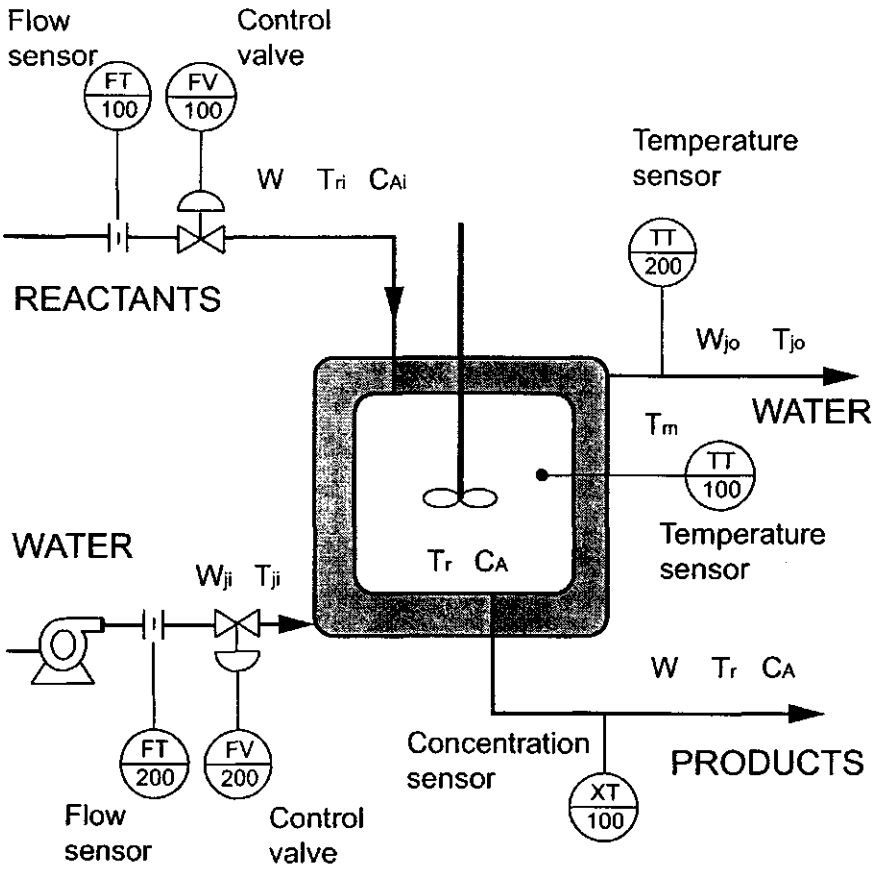

Figure 6.1: Continuous Stirred Tank Reactor process. A second-order exothermic reaction takes place: two components A react irreversibly to form product B. Water flows through the jacket to control the reactor temperature.

We obtain the sampled state-space representation using the continuous state-space representation [Ogata, 1995]. The continuous state-space is generated by the system of linear differential equations, see Appendix E, section E.1.2 for a complete derivation[1].

For the "normal" discrete mode, $z_t = 1$:

$$x_{t+1} = A(z_{t+1})x_t + B(z_{t+1})\gamma_{t+1} + F(z_{t+1})u_{t+1} \quad (6.5)$$

$$y_t = C(z_t)x_t + D(z_t)v_t + G(z_t)u_t \quad (6.6)$$

where:

$$x_t = \begin{bmatrix} C_A(t) \\ T_r(t) \\ T_{jo}(t) \end{bmatrix} \quad y_t = \begin{bmatrix} C_A(t) \\ T_r(t) \\ T_{jo}(t) \end{bmatrix} \quad u_t = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{bmatrix}$$

[1] See [Chen and Howell, 2001] for a similar application.

Table 6.1: CSTR instrumentation. Description.

Tag-name	Functional name	Description
FT-100	Flow sensor/transmitter	Input reactants flow
FT-200	Flow sensor/transmitter	Input water flow
FV-100	Control valve	Reactants flow valve
FV-200	Control valve	Water flow valve
TT-100	Temperature sensor/transmitter	Reactor temperature
TT-200	Temperature sensor/transmitter	Output water temperature
XT-100	Analyzer sensor/transmitter	Output products concentration

Table 6.2: CSTR (4 discrete modes). Operating conditions.

z_t	Model name	Description	Variation
1	Normal	Clean heat transfer area	none
2	Fouling-1	Dirty heat transfer area	5 % fouling
3	Fouling-2	Dirty heat transfer area	10 % fouling
4	Fouling-3	Dirty heat transfer area	15 % fouling

γ_t and υ_t are the process and measurement noises; both follow $\mathcal{N}(0,1)$. The matrices are:

$$A(z_t=1)=\begin{bmatrix} 0.9752 & -0.000214 & -3.55(10)^{-7} \\ 0.2376 & 0.9943 & 0.00325 \\ 0.00176 & 0.01465 & 0.9596 \end{bmatrix} F(z_t=1)=\begin{bmatrix} 0.1299 & 0 & 0 \\ -0.1635 & 0 & 0 \\ 2.0560 & 0 & 0 \end{bmatrix}$$

$B(z_t=1)=n_{process}I^{3\times3}$, $C(z_t=1)=I^{3\times3}$, $D(z_t=1)=n_{measurement}I^{3\times3}$, $G(z_t=1)=0^{3\times3}$. The values for $n_{process}$ and $n_{measurement}$ are fixed for each test. Corresponding results were obtained for each faulty discrete mode ($z_t=2,3,4$); see Appendix E section E.1.2 for details.

10 Discrete modes.

In order to increase the complexity, another 6 possible faulty conditions were considered; see Table 6.3. The equations (JMLG model) for these new discrete modes are also shown in Appendix E section E.1.4.

Note that we do not have to learn the *JMLG* model parameters via the *Expectation-Maximization* algorithm. We simply compute and assign them. The modelling procedure is simplified to three basic steps:

1. Definition of each discrete mode.
2. Linearization around the discrete mode condition.
3. Definition of the measurement and process noises.

Table 6.3: CSTR (10 discrete modes). Operating conditions.

z_t	Model name	Description	Variation
1	Normal	Normal operating conditions	none
2	Fouling-1	Dirty heat transfer area	5 % fouling
3	Fouling-2	Dirty heat transfer area	10 % fouling
4	Fouling-3	Dirty heat transfer area	15 % fouling
5	Concentration-1	Concentration of reactant A	10 % less mass
6	Concentration-2	Concentration of reactant A	20 % less mass
7	Water-1	Water cooling rate at jacket	5 % less water
8	Water-2	Water cooling rate at jacket	10 % less water
9	Jacket-1	Inlet jacket temperature	5 % colder
10	Jacket-2	Inlet jacket temperature	10 % colder

6.2.3 Diagnosis/estimation tests

4 Discrete modes.

Because the linearized model in each discrete mode is only sectionally valid, one must expect some modelling errors. Figure 6.2 shows a simulation of the nonlinear system and compares it to the corresponding linearized *JMLG* model. These simulations were designed using the transition matrix and prior probabilities shown in equation (6.7).

$$P(z_t|z_{t-1}) = \begin{bmatrix} 0.9983 & 0.001 & 0.0005 & 0.00025 \\ 0.001 & 0.9975 & 0.001 & 0.0005 \\ 0.0005 & 0.001 & 0.9975 & 0.001 \\ 0.00025 & 0.0005 & 0.001 & 0.9983 \end{bmatrix} \quad P(z_0) = \begin{bmatrix} 0.9983 \\ 0.001 \\ 0.0005 \\ 0.00025 \end{bmatrix}' \tag{6.7}$$

10 Discrete modes.

Figure 6.3 shows a simulation of the nonlinear system and the corresponding linearized *JMLG* model. Additional random sequences are shown in Appendix E sections (E.1.3-E.1.4).

6.2.4 Results

4 Discrete modes.

Figure 6.4 plots the diagnosis error for the CSTR (4 discrete modes) system. The left graph shows diagnosis error versus number of particles, while the right graph shows diagnosis error versus computing time per time step. Figure 6.5 plots results for a similar test in which the noise level in the measurement signal was increased 100 %. Note the resulting increase in diagnosis error.

To better illustrate the variability in these results, Figure 6.6 again shows diagnosis error for the CSTR (4 discrete modes, normal noise level), but as a box and whisker plot. As we can see, *la-RBPF* outperforms the *PF* and *RBPF* algorithms.

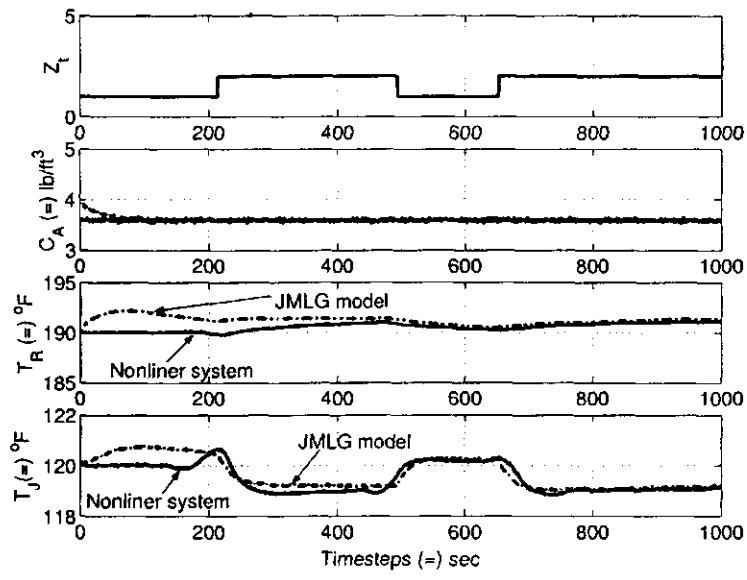

Figure 6.2: CSTR (4 discrete modes). Nonlinear and JMLG simulation. The top plot shows the discrete modes over time. The remaining plots show the corresponding output concentration of A, reactor temperature, and water jacket temperature. The linearized *JMLG* model is also plotted. The main difference between the nonlinear system and the linearized one appears at the beginning.

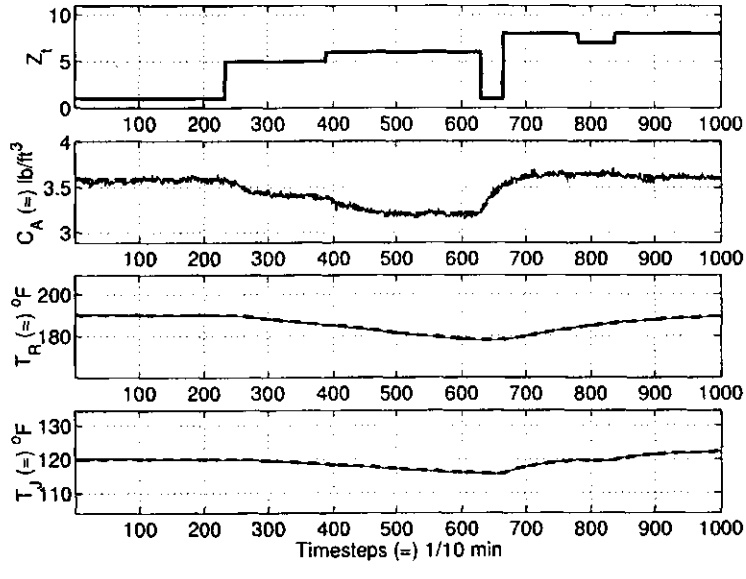

Figure 6.3: CSTR (10 discrete modes). Nonlinear and JMLG simulation. The top plot shows the discrete modes over time. The remaining plots show the corresponding output concentration of A, reactor temperature, and water jacket temperature. The linearized *JMLG* model is also plotted. The main difference between the nonlinear system and the linearized onee appears at the beginning.

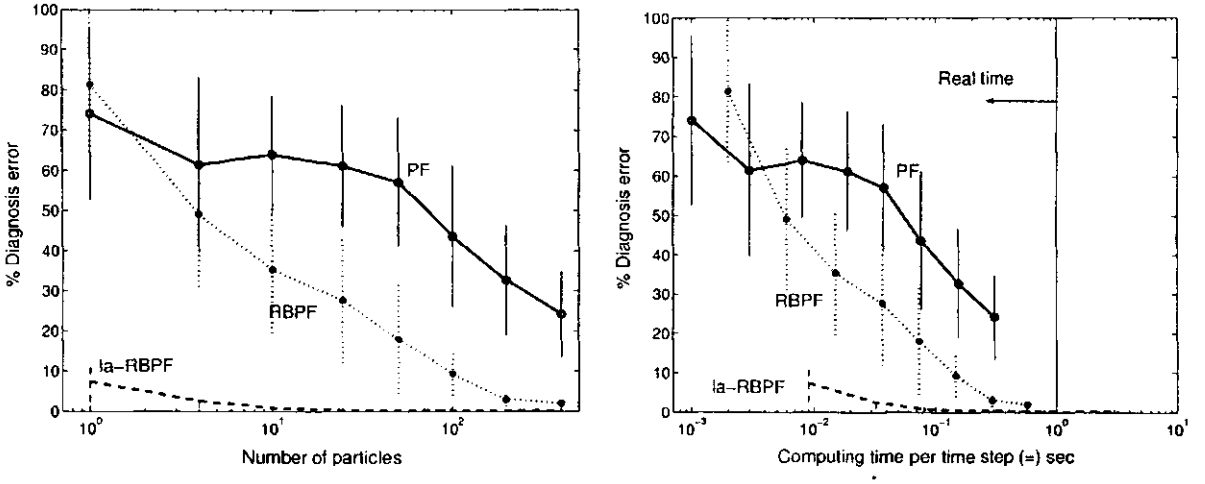

Figure 6.4: CSTR (4 discrete modes). Diagnosis error, low noise level.

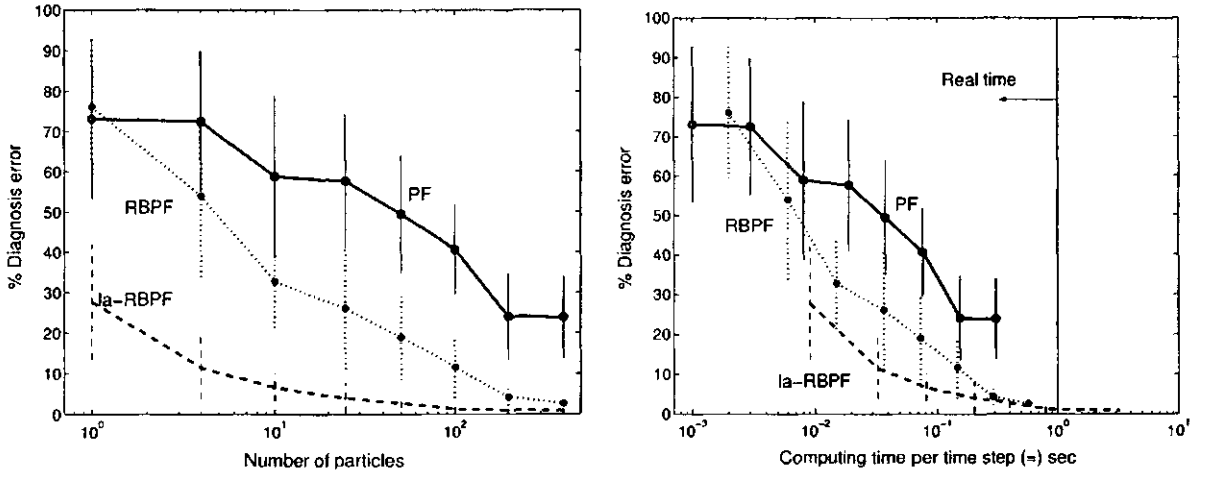

Figure 6.5: CSTR (4 discrete modes). Diagnosis error, high noise level.

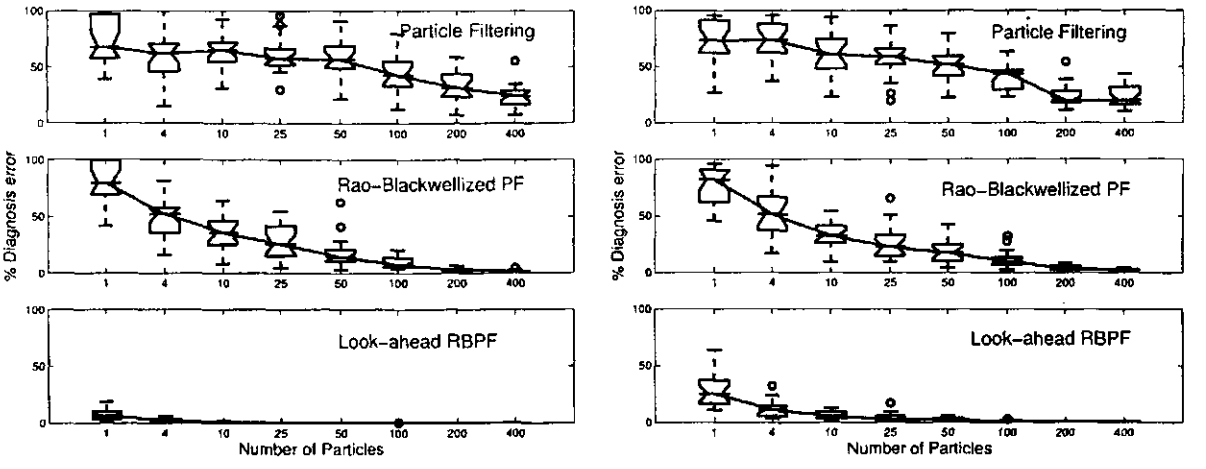

Figure 6.6: CSTR (4 discrete modes). Diagnosis error. Using a box and whisker plot for each Particle Filtering algorithm, we can better appreciate the variability.

10 Discrete states.

Figures (6.7-6.8) show diagnosis error for the CSTR (10 discrete modes) system. The left plots show diagnosis error versus number of particles, while the right plots show diagnosis error versus computing time per time step. Figure 6.8 was generated using a measurement signal 100 % noiser than that in Figure 6.7.

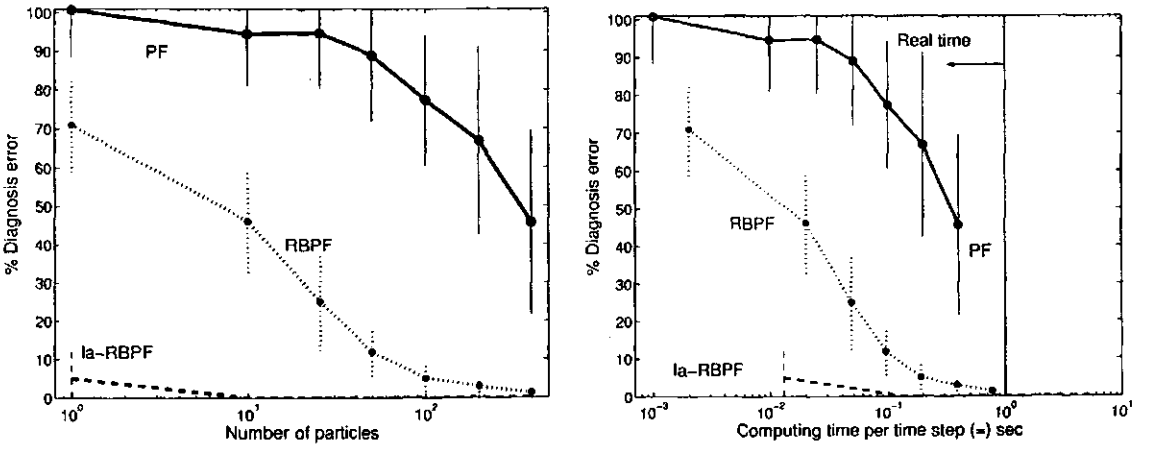

Figure 6.7: CSTR (10 discrete modes). Diagnosis error, low noise level.

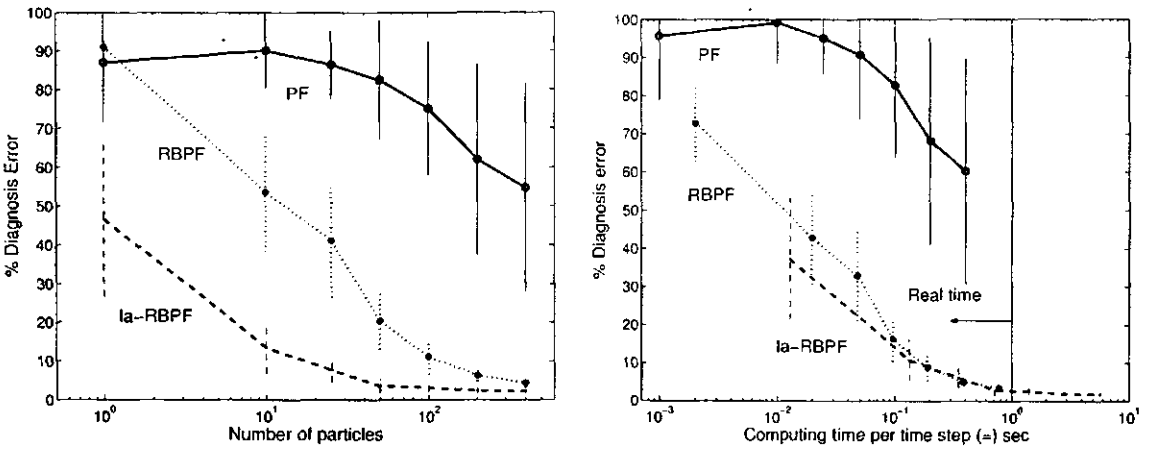

Figure 6.8: CSTR (10 discrete modes). Diagnosis error, high noise level.

6.3 Simulated Mobile Robot

In order to evaluate how the number of discrete modes affects the inference process, we combined our two real JMLG models for the mobile robot (smooth and tiled floor) into a single model. We also played with the process and measurement noises.

6.3.1 Modelling

8 Discrete modes.

Table 6.4 shows the complete JMLG model parameters for the 8 discrete mode system. We defined the process noise n_p, and chose the measurement noises $n_m = n_p, 2n_p, 4n_p$.

Table 6.4: Simulated Mobile Robot. JMLG parameters.

z_t	$x_0(z_t)$	$A(z_t)$	$B(z_t)$	$C(z_t)$	$D(z_t)$	$F(z_t)$	$G(z_t)$
1	0.5619	0.7298	n_p	1	n_m	0.1518	0
2	0.5037	0.7385	n_p	1	n_m	0.1317	0
3	0.4265	0.6926	n_p	1	n_m	0.1311	0
4	0.3581	0.7491	n_p	1	n_m	0.0898	0
5	0.5240	0.6101	n_p	1	n_m	0.2043	0
6	0.4645	0.7032	n_p	1	n_m	0.1379	0
7	0.4028	0.8194	n_p	1	n_m	0.0727	0
8	0.3179	0.8430	n_p	1	n_m	0.0499	0

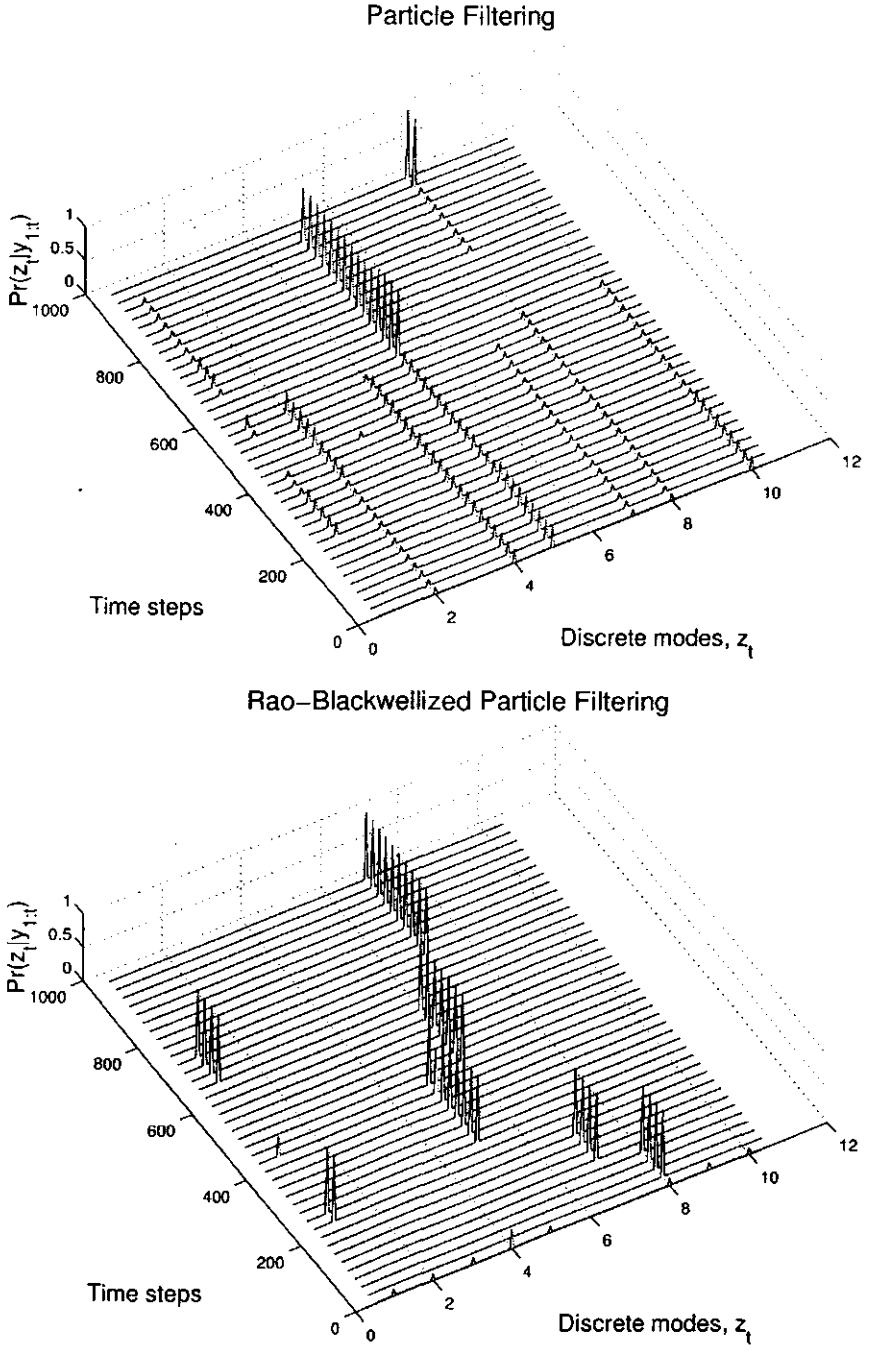

Figure 6.9: CSTR (10 discrete modes). Probability distribution $p(z_t|y_{1:t})$. The upper plot shows $p(z_t|y_{1:t})$ as approximated by the standard *PF* algorithm; the lower plot shows *RBPF*. Both approximations used 10 particles.

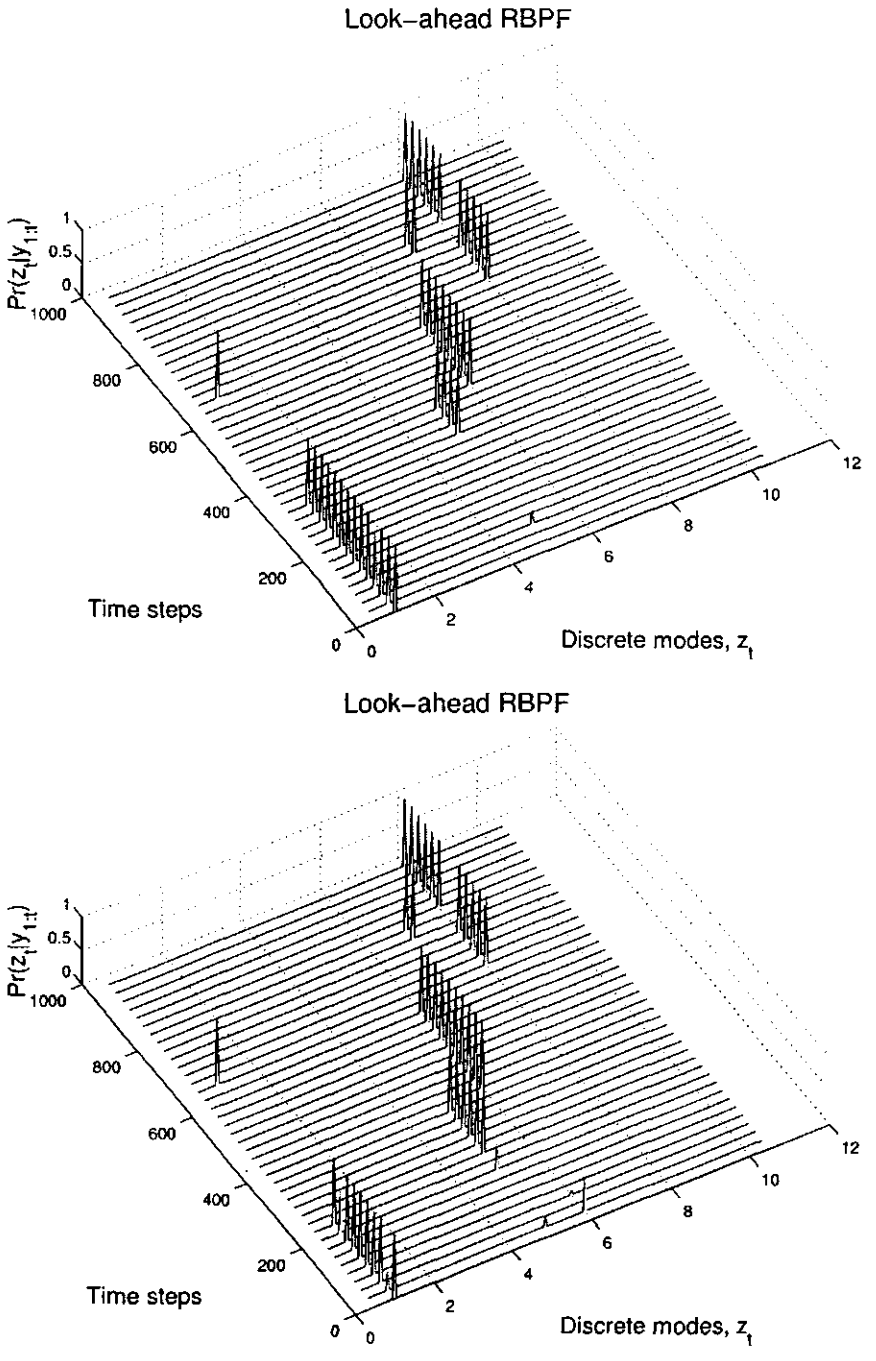

Figure 6.10: CSTR (10 discrete modes). Probability distribution $p(z_t|y_{1:t})$. Both plots show $p(z_t|y_{1:t})$ as approximated using *la-RBPF*. The upper plot used 10 particles; the lower plot used 1,600 particles.

18 Discrete modes.

We also tested the Particle Filtering algorithms with an 18 discrete mode JMLG model. We defined this system as follows:

- Three kind of floor: smooth (low noise level), tiled (medium noise level), and terrain (high noise level).
- For each floor, six different loads, resulting in different speeds.

For the smooth floor, we have $z_t = \{1, \ldots, 6\}$ where $z_t = 1$ runs faster than $z_t = 6$. For the tiled floor $\left(z_t = \{7, \ldots, 12\}\right)$, $z_t = 7$ runs faster than $z_t = 12$. For the terrain floor $\left(z_t = \{13, \ldots, 18\}\right)$, $z_t = 13$ runs faster than $z_t = 18$.

6.3.2 Diagnosis/estimation tests

8 Discrete modes.

Several random sequences were generated using the transition matrix and initial prior probabilities shown in equation (6.8).

$$P(z_t|z_{t-1}) = \begin{bmatrix} 0.99 & 0.005 & 0.0005 & 0.00025 & 0.00425 & 0.0 & 0.0 & 0.0 \\ 0.005 & 0.9895 & 0.005 & 0.0005 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0005 & 0.005 & 0.9895 & 0.005 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.00025 & 0.0005 & 0.005 & 0.99425 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.00005 & 0.0 & 0.0 & 0.0 & 0.9942 & 0.005 & 0.0005 & 0.00025 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.005 & 0.9895 & 0.005 & 0.0005 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0005 & 0.005 & 0.9895 & 0.005 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.00025 & 0.0005 & 0.005 & 0.99425 \end{bmatrix} \quad (6.8)$$

$$P(z_0) = \begin{bmatrix} 0.9825 & 0.01 & 0.005 & 0.0025 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Figure 6.11 shows two representative sequences. The upper plots show the discrete mode over time; the lower plots show the mobile robot's speed. For the right graph we used a noisier measurement signal. Appendix E section E.2.1 shows another random sequence.

18 Discrete modes

Figure 6.12 shows two representative random sequences. The upper plots show the different discrete modes that the robot was in. Discrete modes 1-6 correspond to smooth floor, 7-12 to tiled floor, and 13-18 to terrain floor. The lower plots show the robot's resulting speed. In the left graph, the robot spent a lot of time on the smooth floor, typically a low noise level condition. The right plot mainly involves the tiled floor, which has a medium noise level.

Appendix E section E.2.1 shows another random sequence for both groups of discrete modes.

6.3.3 Results

8 Discrete modes.

Figures (6.13-6.14) show diagnosis error versus number of particles, and diagnosis error versus computing time per time-step, respectively, for the simulated mobile robot (8 discrete modes).

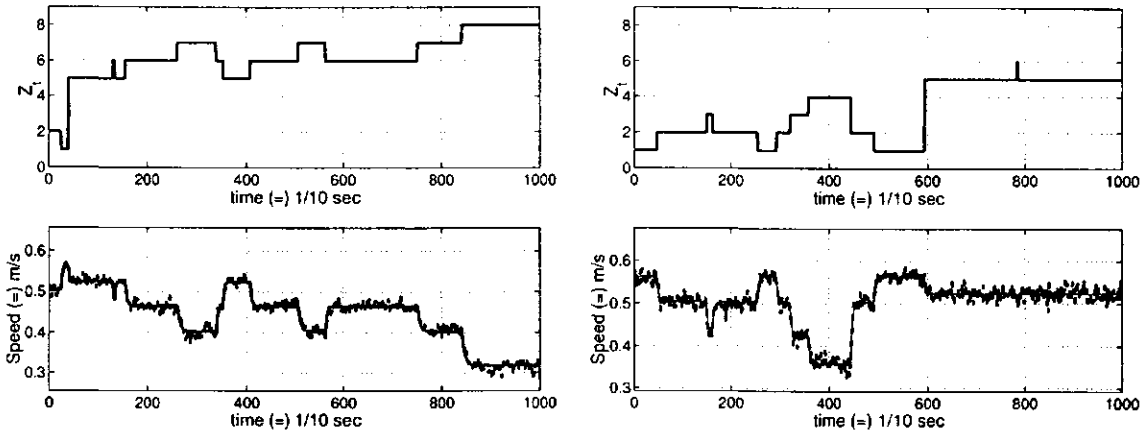

Figure 6.11: Mobile robot (8 discrete modes). Random sequences. The upper plots show the discrete mode over time; the lower plots show the resulting speed as simulated by the *JMLG* model. The right plots were implemented using a higher noise level.

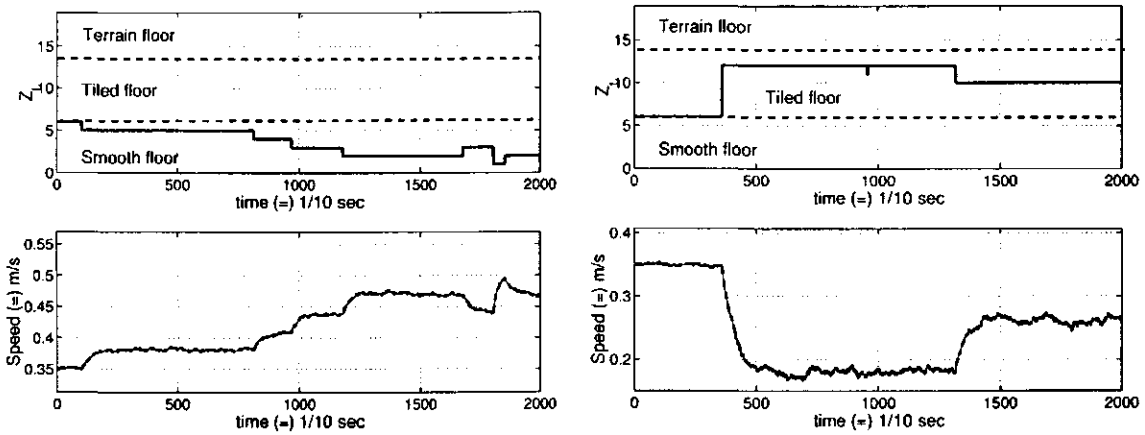

Figure 6.12: Mobile robot (18 discrete modes). Random sequences. The upper plots show the discrete modes, and the lower plots show the robot speed simulated by the *JMLG* model. The discrete modes are classified into smooth, tiled and terrain floor. In the left graphs the robot mainly walks on the smooth floor; the right graphs mostly involve the tiled floor.

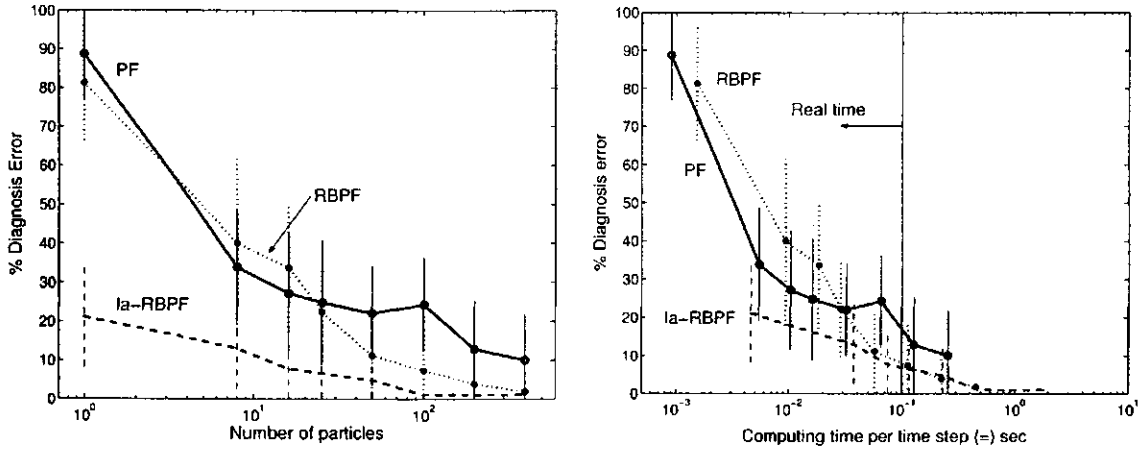

Figure 6.13: Mobile robot (8 discrete modes). Low noise level. The left plot shows diagnosis error versus number of particles, while the right plot shows diagnosis error versus computing time. The noise level in the measurement signal was low.

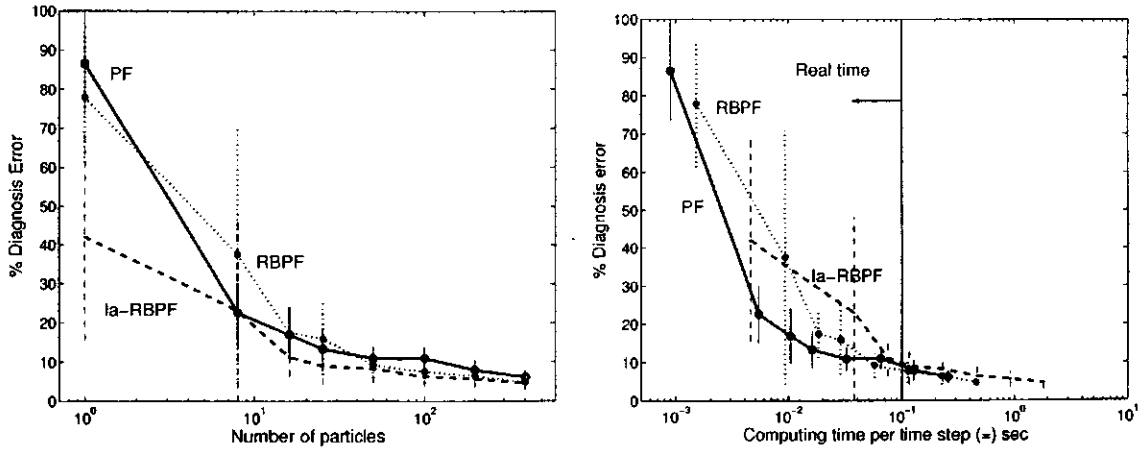

Figure 6.14: Mobile robot (8 discrete modes). High noise level. The left plot shows diagnosis error versus number of particles, while the right plot shows diagnosis error versus computing time. The noise level in the measurement signal was high.

Figure 6.15 shows a box and whisker plot of the above results. The left graphs correspond to a low noise level, and the right graphs to a high noise level.

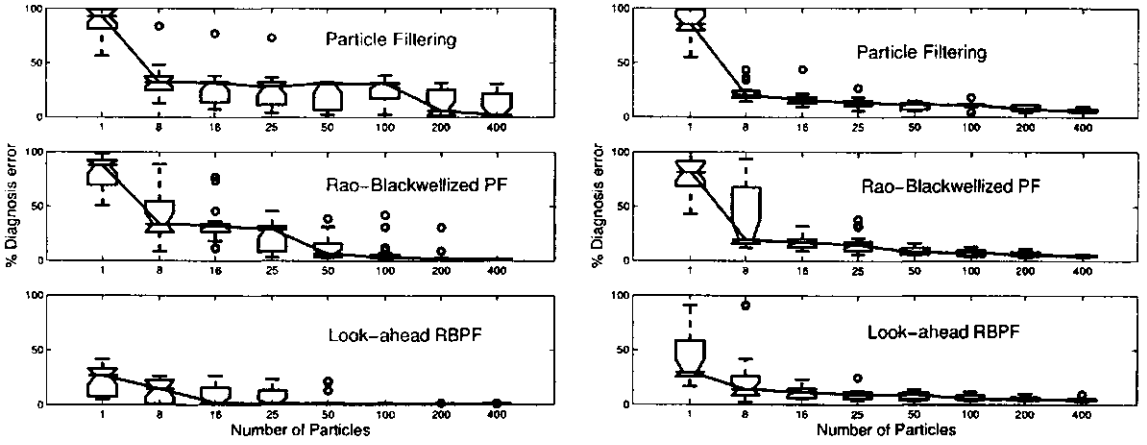

Figure 6.15: Mobile robot (8 discrete modes). Comparison. Box and whisker plots show the diagnosis error versus number of particles for the three Particle Filtering algorithms. The left plots correspond to a low noise level while the right plots represent a high noise level.

18 Discrete modes

Figures (6.16-6.17) show diagnosis error versus number of particles, and diagnosis error versus computing time per time-step, respectively, for the simulated mobile robot (18 discrete states).

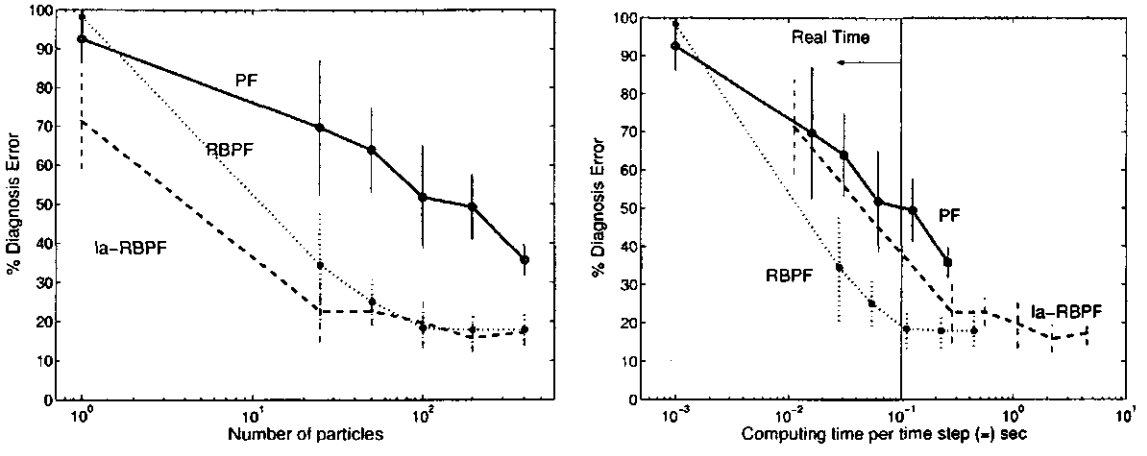

Figure 6.16: Mobile robot (18 discrete modes). High noise level.

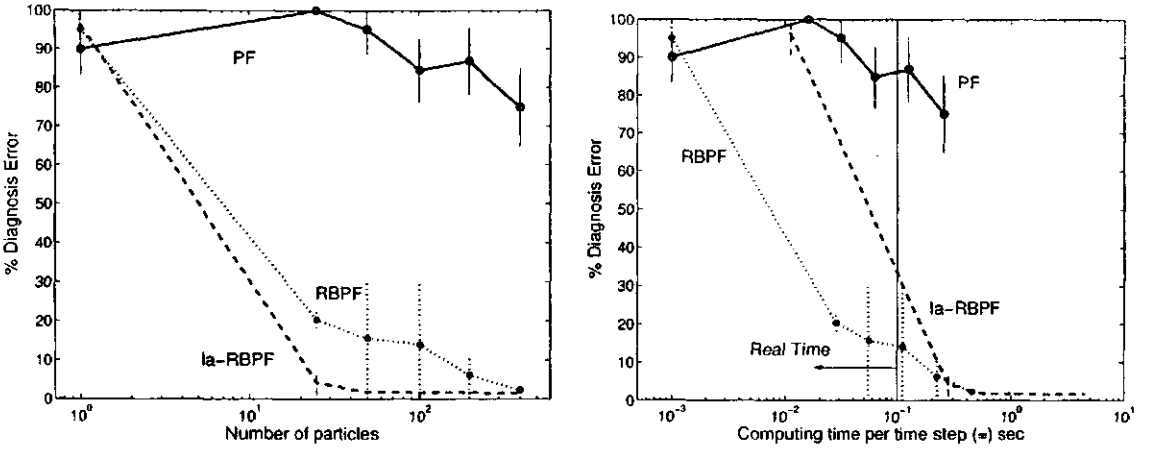

Figure 6.17: Mobile robot (18 discrete modes). Low noise level.

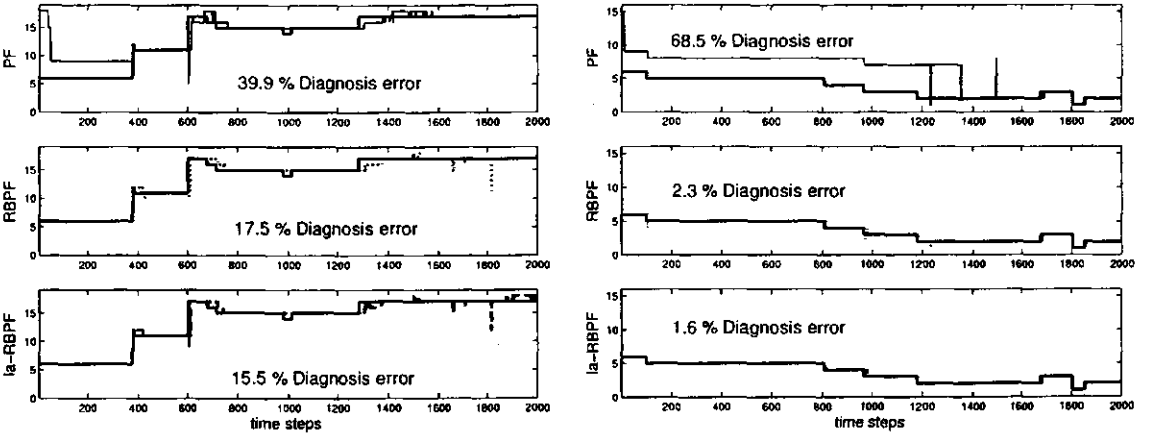

Figure 6.18: Mobile robot (18 discrete modes). MAP estimation comparison. The left graphs correspond to high noise level, while the right graphs correspond to low noise level.

6.4 Summary

We worked with two simulated domains, the Continuous Stirred Tank Reactor (*CSTR*) and an extended version of the mobile robot system. We compared three Particle Filtering algorithms in these domains, testing greater numbers of discrete modes, more continuous state space variables, and different proportions of measurement vs. process noise.

Preliminary results indicate that all three Particle Filtering algorithms can work in these conditions; however, as the number of discrete modes grows, *RBPF* and especially *la-RBPF* require excessive computing time per time step. If such domains demand quick diagnosis, *la-RBPF* will be limited to working off-line.

If one only compares the results based on number of particles, *la-RBPF* always outperforms *PF* and *RBPF*. *la-RBPF* gives lower diagnosis error and lower variance.

Generally, noisy signals[2] have an adverse effect on Particle Filtering algorithms. In this case, *la-RBPF* performs extra processing with no benefit. In a high noise level system, *PF* and *RBPF* perform better than *la-RBPF* in terms of computing time per time step. We could overcome this situation using parallel computing, but this would affect operating costs.

We can conclude that *la-RBPF* works well in these domains, but it could be restricted to off-line operation when the number of discrete modes grows.

[2]We are assuming that the noisy signals cannot be filtered.

Chapter 7

Conclusions and Future Work

7.1 Introduction

In this chapter, we compare, discuss and summarize the main results we obtained for the experimental domains (Chapter 5) and simulated systems (Chapter 6) using the new *la-RBPF* algorithm (Chapter 4). Our discussions cover both the *JMLG* model and the *la-RBPF* algorithm. Based on these discussions, we derive our conclusions. We divide our conclusions into main contributions and limitations. Finally, possible future directions for this research are proposed.

7.2 Related work

Fault diagnosis for autonomous operation systems such as spacecraft and planetary rovers demands efficient and any time algorithms. As we discussed, Particle Filters have a number of properties that make a practical algorithm for diagnosis in this domain. Specifically, it is very attractive for planetary rovers because their anytime properties.

[Hutter and Dearden, 2003b] introduced the *Gaussian Particle Filter* (*GPF*), an efficient variant of the Particle Filtering algorithms for non-linear hybrid systems. Like *la-RBPF* [Morales-Menéndez *et al.*, 2002], GPF_2 achieves improved performance by:

- Sampling directly from the posterior distribution, and
- Resampling before the transition

The new improved algorithm GPF_2 only differs from *la-RBPF*[1] in that is calling an unscented Kalman filter updated [van der Merwe *et al.*, 2001] instead of Kalman filter updated.

Some experiments on a suspension system of the *K-9* planetary rover at NASA Ames research center show impressive results. The *K-9* planetary rover is a six-wheeled rover with a rocker-bogey suspension, Figure 7.1. The suspension's response to driving over rocks and other obstacles was modelled to anticipate faulty situations. The model has six discrete modes and six continuous variables, with non-linear dynamics in three discrete modes, two of which are observable.

Experiments were performed on a simple model of *K-9*. Graphs similar to ours show that GPF_2 (Gaussian Particle Filter based on *la-RBPF*'s features) outperforms standard Particle filters in several ways, such as:

[1] Originally named $RBPF_2$ during its testing phase

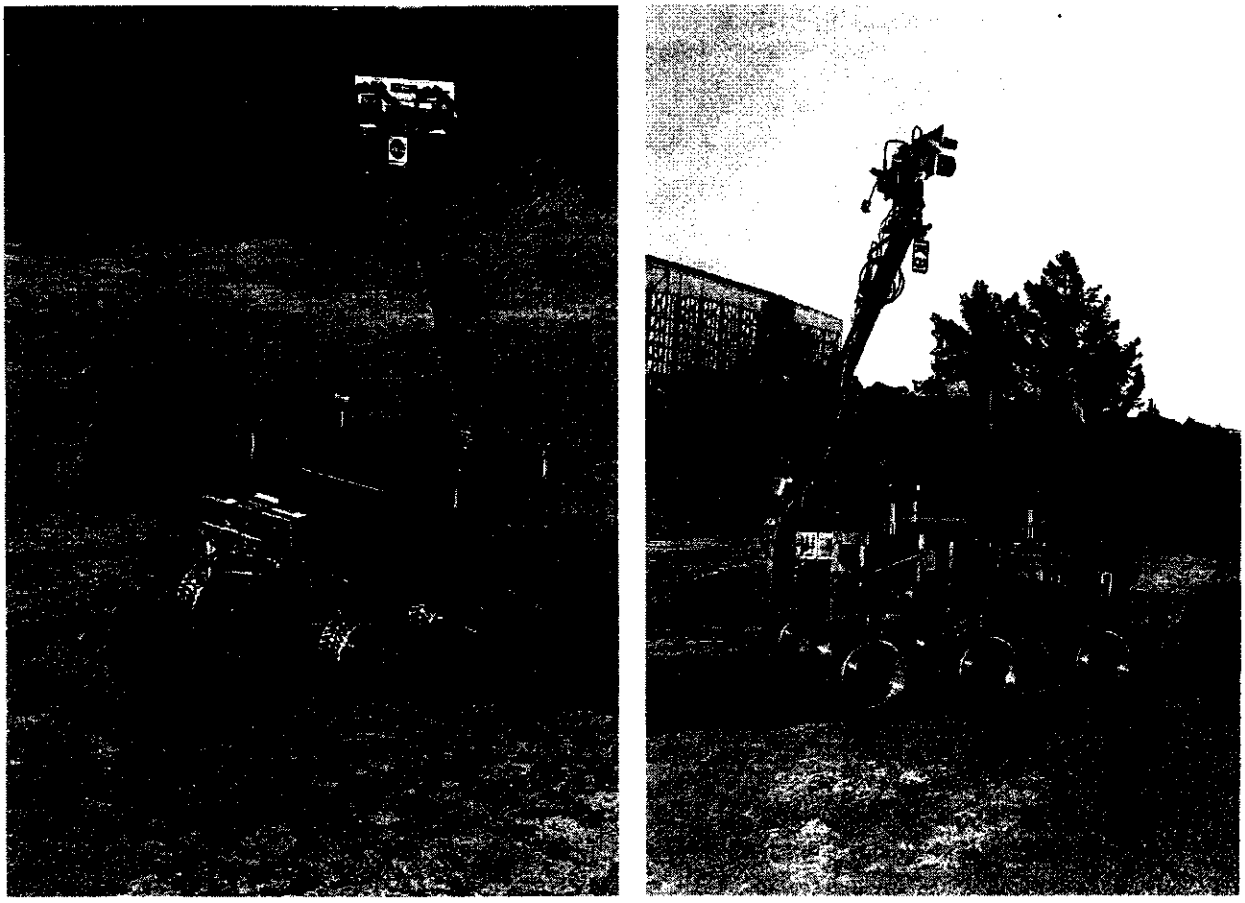

Figure 7.1: K-9 planetary rovers. K-9 is a six-wheeled rover with a rocker-bogey suspension. *Pictures provided by Tom Trower, NASA Ames Research Center.*

- Diagnosing continuous parameters of hybrid systems.
- Achieving a significantly lower *mean square error* (MSE) for the continuous parameters.

Based on real data from the *K-9* planetary rover, GPF_2 successfully identifies all the discrete modes before the standard Particle Filter does. See [Hutter and Dearden, 2003a] for a extended version of this application.

Our approach was based on previous works of De Freitas [Doucet *et al.*, 2000a; de Freitas, 2001] in different domains. Other applications of Particle Filtering in diagnosis were discussed in Chapter 2, section 2.6; however, their domains and modelling frameworks are different from ours.

7.3 Discussions

7.3.1 Jump Markov Linear Gaussian model

The proposed *JMLG* learning algorithm generated good results for each experimental domain (see Chapter 5):

- Industrial dryer, Figure 5.4

- Level tank, Figures 5.10 and 5.11
- Heat exchanger, Figure 5.17
- Mobile robot, Figure 5.24

The above graphs show real and synthetic data. The synthetic data were generated using the *JMLG* model. For clarity these graphs omit the noise matrices (i.e. $B = 0$ and $D = 0$).

We validated the *JMLG* model by following this procedure:

1. Simulate the Markov chain
2. Physically implement the discrete modes (e.g. faults or operating conditions) in the process
3. Collect input/output variables
4. Simulate the *JMLG* model under same physical conditions
5. Compare the measurement and synthetic data

We could use a formal comparison procedure such as residual analysis (Appendix C, section C.1.4); however, we are not looking for the *best* dynamic representation; we want a *good* model in order to diagnose/estimate the process. Our comparison was mainly visual analysis based on some statistics such as variance and sum-of-squares errors, plus a lot of experimental work (more than 30 tests on average per domain).

Our main comments about the *JMLG* model shown in Chapter 5 and Appendix D are:

- The *JMLG* model successfully represents the dynamic behaviour of the process. For each discrete mode the synthetic data and observations have the same general behaviour (increasing/decreasing).
- For some processes, the *JMLG* model presented a small difference in the final steady value. Figure 7.2 (same Figure 5.10) shows this difference as an *Error*.

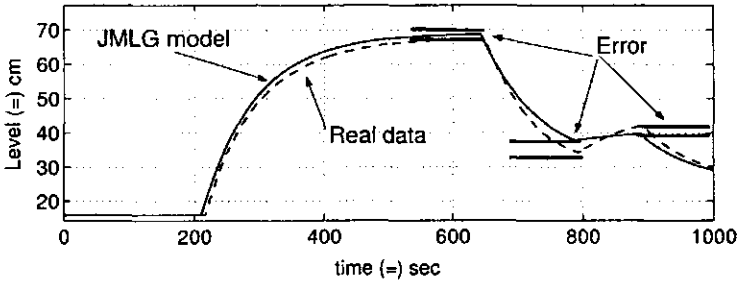

Figure 7.2: Level tank. Comparison between real data and the *JMLG* model. The shown error is mainly generated because the non-linear characteristics of the process.

This difference occurs because we are trying to capture the non-linear behaviour of this process using linear models. The absolute global change in the level tank when we open the input flow valve $X\%$ is different to the absolute global change in level when we close the input flow valve $X\%$. The nonlinear actuator in the control valve explains this situation. We have the same situation with the heat exchanger. We can solve this problem if we break the process into more linear models, however this unnecessarily increases the number of discrete modes. The original model gives good practical results.

- The initial conditions sometimes were not properly estimated. Figure 7.3 (same Figure 5.11) shows this difference as an *Error*.

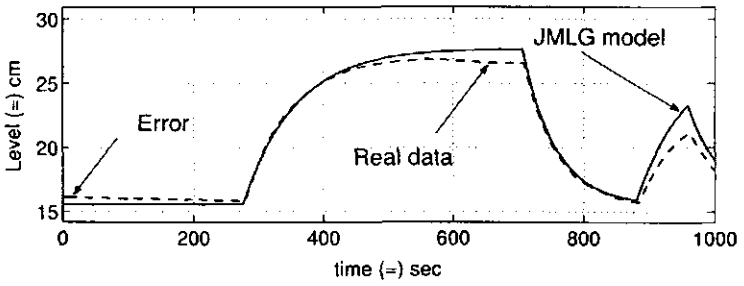

Figure 7.3: Level tank. Modelling error in initial conditions. There is a small difference between the real and synthetic data at $t = 0$.

However, this error has no significant impact. The error occurs because it is very difficult to keep the same operating conditions during the experimental tests (e.g. the steam thermodynamic properties for the heat exchanger, or the ambient air conditions for the industrial dryer).

- Dead time is the elapsed time between the application of an action and the process reacting. Sometimes it was difficult to estimate, as Figure 7.4 shows (same Figure 5.4). Often, this difference was also generated by human error in timing the state changes. This error is higher for runs with more state changes, especially for the mobile robot where the sampling rate is 0.1 sec.

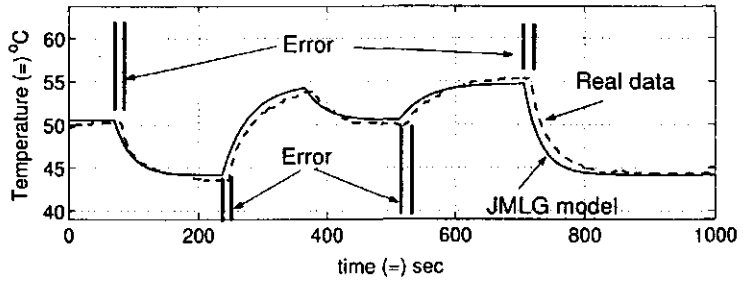

Figure 7.4: Industrial dryer. Error in dead time estimation. Time delay in process is caused by transportation lag, a phenomenon also commonly known as dead time. Dead time is the elapsed time between the application of an action and the process reacting. Dead time has a negative effect in control systems.

- For each domain, we learned the *JMLG* model parameters several times with different training data. Then we evaluated the model using additional test data. We found that the training data needs to adequately represent the transient response. If too much data from steady conditions is used, the *JMLG* model parameters become more representative of steady state conditions and we lose the transient dynamic information.

7.3.2 Look-ahead Rao-Blackwellized Particle Filtering algorithm

The *PF*, *RBPF* and *la-RBPF* algorithms were tested with real industrial applications (Chapter 5) and simulated processes (Chapter 6). Basically, given the observable data $\{u_t, y_t\}_{t=1}^{T}$ the inference algorithms must estimate the most probable hybrid state, where the most important variable is the discrete mode $\{\widehat{z}_t\}_{t=1}^{T}$. We can compare this estimate with the true value, $\{z_t\}_{t=1}^{T}$, and generate a basic performance characterization : *diagnosis error.*

The diagnosis error is the percentage of time steps during which the discrete mode was not identified properly. We computed the average and standard deviation of this indicator over 25-30 independent runs for different number of particles. Usually, each run has 1,000 time steps. The results are represented in two graphs:

1. *Diagnosis error versus number of particles.* We plotted this graph starting with one particle, then n_z (maximum number of discrete modes) particles, then 100 % more particles each time. The diagnosis error scale is decimal and the number of particles scale is logarithmic.

2. *Diagnosis error versus computing time.* Each algorithm demands different computing time per particle, so we also plotted diagnosis error versus computing time for the same numbers of particles as the first graph.

Based on these graphs for each domain we find the following features:

- *Diagnosis error versus number of particles.*
  For the three Particle Filtering algorithms the diagnosis error decreases as the number of particle increases. Usually, *PF* and *RBPF* show the same diagnosis error for one particle, but as the number of particles increases the diagnosis error for the *RBPF* algorithm decreases more quickly than for *PF*. As we can see, for all the graphs the proposed *la-RBPF* algorithm always works significantly better than the regular *PF* and *RBPF*. *la-RBPF* gives a very low diagnosis error per number of particles.

  Figure 7.5 (same information in Figure 5.18) shows a separate box and whisker plot for each Particle Filtering algorithm for different numbers of particles. The boxes have lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of a box to show the extent of the rest of the data. The boxes are notched. Notches represent a robust estimate of the uncertainty about the medians for box-to-box comparison. Outliers are data values beyond the ends of the whiskers; outliners are shown as circles. Note how the *la-RBPF* quartiles are closer than the *PF* and *RBPF* quartiles.

- *Diagnosis error versus computing time.*
  The number of computational steps per particle is different for each algorithm, so their computing times are different. *la-RBPF* is the most expensive algorithm, *RBPF* is the second most expensive algorithm and *PF* is the cheapest one. However, *la-RBPF* showed lower diagnosis error than *PF* and *RBPF* per unit of computing time. Standard *RBPF* gave lower diagnosis error than *PF*.

- *Prior probabilities of faults.*
  Fault diagnosis is a difficult task in general because the prior probabilities of faults usually are very low. The task is more difficult when we use approximation methods such as particle filtering because particle generation is strongly based on these prior probabilities. We can get high diagnosis error results despite the observations. As we show in Chapter 4, *la-RBPF* can generate samples of the discrete modes z_t based on the true posterior distribution. The posterior distribution captures the evidence every time step, so if the observations show faulty conditions, *la-RBPF* can identify the faulty states despite the low prior probabilities.

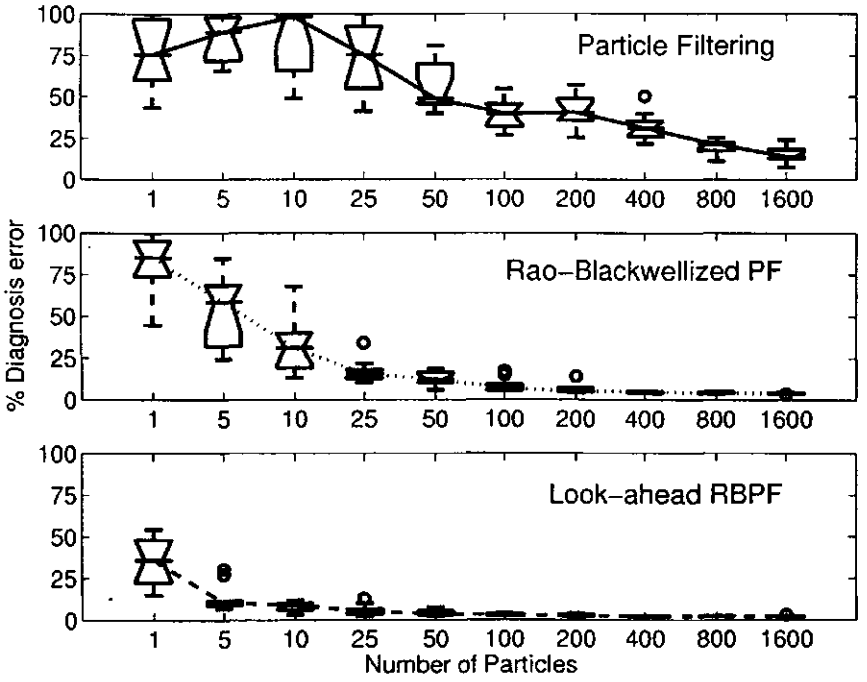

Figure 7.5: Box and whisker plots for each *PF* algorithm. The *la-RBPF* algorithm always works significantly better than the regular *PF* and *RBPF* for each number of particles.

- *Real time applications.*
  On average, we consider 1 sec to be the borderline between real time and off line diagnosis. This is a base reference only. For some domains, such as the heat exchanger, 1 sec is too short; we can make decisions and take action every 2 sec. However, for the industrial dryer, 1 sec is too long; this process has a faster time response and we have to take actions faster too. If we have limited computational resources or a specified computing time, *la-RBPF* provides lower diagnosis error. For example, in the level tank domain (Figure 5.12), if we are limited to 0.5 sec computing time, *la-RBPF* produces a 2 % diagnosis error, *RBPF* a 25 % diagnosis error, and *PF* a 35 % diagnosis error.

- *Diagnosis error variance.*
  la-RBPF shows lower variance than standard *PF* and *RBPF* per number of particles. This advantage, based on the Rao-Blackwell formula, grows as the number of particles is increased. As we can see in Figure 7.5, taking 50 particles as an example, *la-RBPF* shows closer quartiles than *PF* and *RBPF*. This feature is very important for stochastic algorithms.

- *Number of discrete modes n_z.*
  If the number of discrete modes grows, the *la-RBPF*'s advantage in diagnosis error per number of particles remains practically unchanged. However, the diagnosis error versus computing time plot is strongly affected. Computing the importance weights in this algorithm requires us to evaluate the following term $\mathcal{N}\big(\mathbb{E}(y_t|y_{t-1}), cov(y_t|y_{1:t-1})\big)$ for each discrete mode.

  Originally, we were looking for an efficient algorithm that works in a distributed environment. Specifically,

an industrial process would be broken into small sections or areas to keep the number of discrete modes n_z small. Here, *la-RBPF* would be an excellent distributed alternative. However, if we cannot keep the number of discrete modes small and we have to work on line, there are other possible solutions such as:

- *Parallel computing.* Particle Filtering algorithms are easily implemented in parallel.
- *Better software.* The three *PF* algorithms were tested using Matlab v 6.0, because of its mathematical and graphical tools; other software like *C* would be more efficient.

- *Types of discrete modes.*
  The number of discrete modes can grow for two main reasons:
  - *Qualitative models.* When different operating conditions or faults appear for the process, and the dynamic behaviour is qualitatively different.
  - *Discretization.* For a specific qualitative model, we have to break the process into several simple models in order to work with non-linear systems.

  We tested the *PF* algorithms with a small number of different qualitative models for the experimental domains due to physical limitations. However, we were able to test discretization by building a non-linear CSTR simulator (Chapter 6, section 6.2). We tested the algorithm with 10 discrete modes, observing similar results and limitations. We also found similar results for the simulated mobile robot with 18 discrete modes (Chapter 6, section 6.3).

- *Number of continuous states n_x.*
  If the number of continuous states n_x grows, we have problems similar to those when n_z grows. The number of continuous states has a direct effect on two important steps of the *la-RBPF* algorithm:
  - Computing the data likelihood for the importance weights (here the effect is multiplicative with n_z)
  - Kalman updating

  where there is a matrix inversion computation which depends on the number of continuous states. This computation represents almost 35 % of the total computing time. However, we can apply solutions similar to those for the n_z-problem.

- *Noise environment.*
  If the signals are very noisy, such as the mobile robot walking on the tiled floor (see Figure 5.24), looking ahead a single time step does not tell us much about the dynamic behaviour, especially at high sampling rates. For the mobile robot domain the sampling rate was 0.1 sec.

  Figure 7.6 shows diagnosis error when the mobile robot walks on a tiled floor. In general, the three Particle Filtering algorithms present similar patterns to previous plots. The upper left graph in Figure 7.6 shows diagnosis error versus number of particles. In this graph *la-RBPF* still gives lower diagnosis error than *PF* and *RBPF*. The lower left graph measures *la-RBPF*'s improvement[2] in diagnosis error over standard *RBPF*, but this improvement is significant only for fewer particles. As the number of particles increases the improvement disappears. Furthermore, the computing time that *la-RBPF* requires for this improvement is not justified. Taking 10 particles as an example, *la-RBPF* shows 10 % improved diagnosis error, but demands 450 % more

[2]Improved diagnosis error = $Error_{RBPF}$ - $Error_{la-RBPF}$

computing time. When we plot diagnosis error versus computing time (right graph in Figure 7.6), *la-RBPF* shows poor performance. *la-RBPF* has higher diagnosis error than *PF* and *RBPF*. In this case the look-ahead is just extra work for insufficient reward.

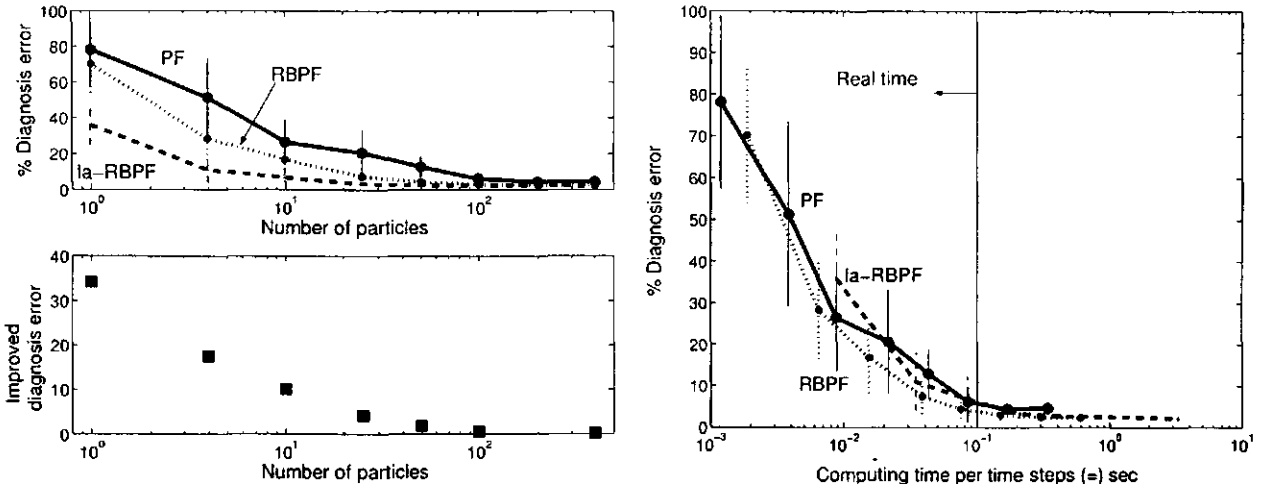

Figure 7.6: Mobile Robot. Diagnosis error on tiled floor. Tiled floor environment has very noisy signals. High noise level represents a traditional problem for Particle filters that requires enormous number of particles to cope with. The upper left graph shows diagnosis error versus number of particles, while lower left graph measures *la-RBPF's* improvement in diagnosis error over standard *RBPF*. The right graph shows diagnosis error versus computing time per time steps for this noisy domain.

In this environment, *la-RBPF*'s only remaining advantage is its low variance, as Figure 7.7 shows. Using *la-RBPF* we get less variance for 25 particles than *RBPF* does with 50 particles.

We can apply a moving-average filter to the raw data[3] in order to get a practical solution. Figure 7.8 shows the raw and the filtered signals.

Using the filtered signal, *la-RBPF*'s advantage is restored, as Figure 7.9 shows. The overhead of the extra processing per particle is once again more than compensated for by the decrease in error.

We tested this hypothesis further via some simulations using the *JMLG* model for the mobile robot (18 discrete modes). As we can see, when the mobile robot walks in high level noise environments, Figure 6.16, *la-RBPF* has limited advantage. For low level noise environments, Figure 6.17, its advantage is recovered.

- *Tracking performance.*
  Figure 7.10 shows the tracking performance for the heat exchanger domain. The upper subgraphs in both plots represent the true discrete mode of the process. The left graphs show the MAP estimation for each algorithm using 50 particles, while the right graphs show the same information for 1,600 particles. Note *la-RBPF* gets the same percentage of diagnosis error with 50 particles that standard *PF* gets with 1,600 particles.

  Even with 1,600 particles, all three algorithms in Figure 7.10 show high variability in the maximum a posterior (MAP) estimation between time steps 350^{th} and 500^{th}. This variability occurs because the algorithms are

[3]We did not perform a formal signal processing analysis

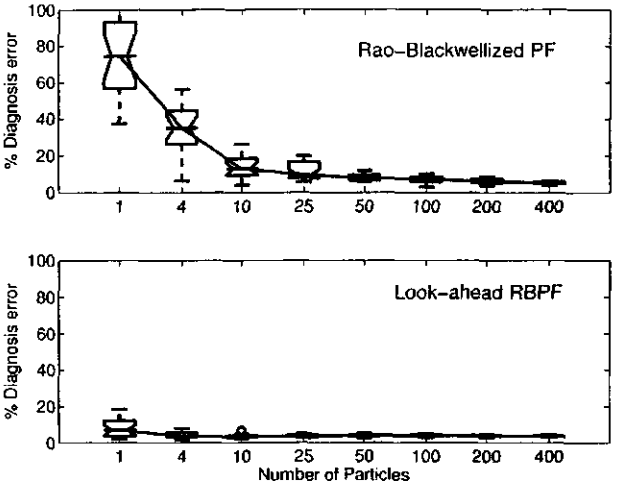

Figure 7.7: Mobile robot. Diagnosis error variance on noisy environment. Box and whisker plots show *la-RBPF*'s advantage in variance and median reduction per number of particles.

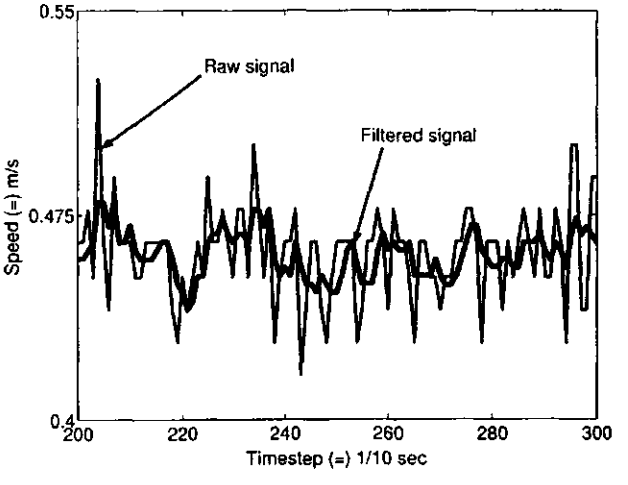

Figure 7.8: Mobile robot. Raw and filtered signal. A moving-average filter was used for the noisy raw signal.

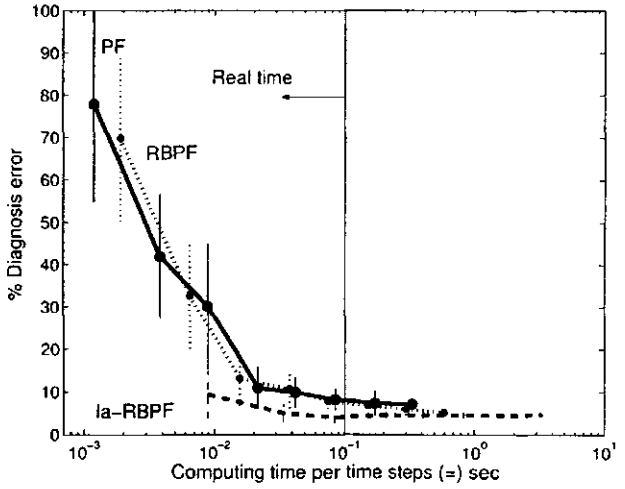

Figure 7.9: Mobile robot. Diagnosis error using the filtered signal. *la-RBPFs* advantage is restored. *la-RBPF* shows lower diagnosis error for a specific computing time. Note, it has good performance for the real-time side.

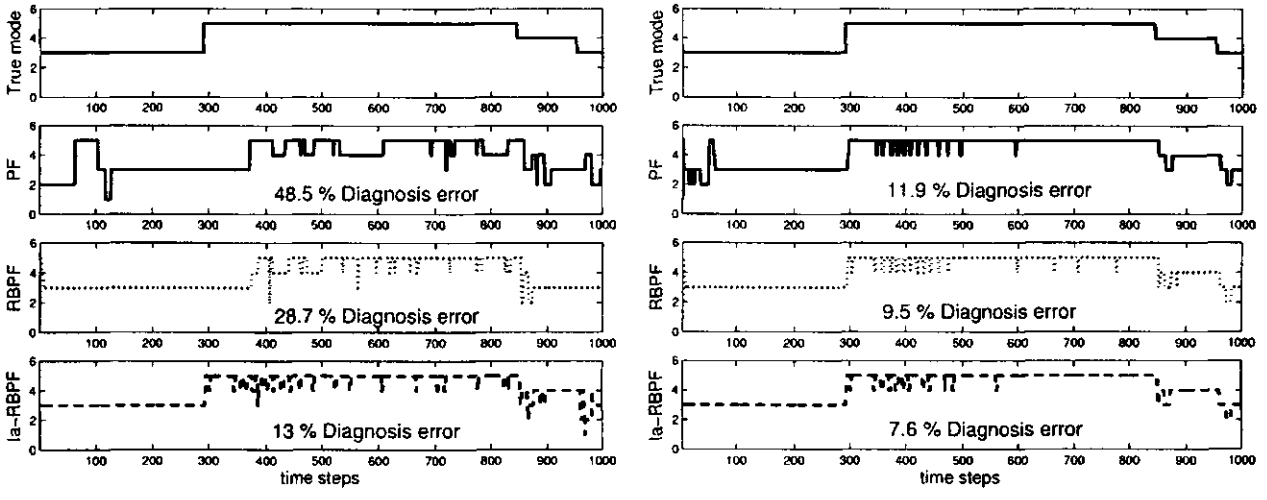

Figure 7.10: Heat exchanger. Tracking performance. Left graphs show the maximum a posteriori(MAP) estimation for each algorithm using 50 particles, while the right graphs shows the same information for 1,600 particles. Each graph includes the percentage of diagnosis error based on the true mode shown in the upper graphs.

diagnosing the process mode during transient behaviour. Figure 7.11 shows the heat exchanger data used during this estimation. The lower graph shows how the temperature changes for the different discrete modes (upper graph) during this period. Note that after the 500^{th} time step the process is almost in steady state and the MAP estimation is better. Performance for the discrete mode changes between the 800^{th} and $1,000^{th}$ time steps is similar.

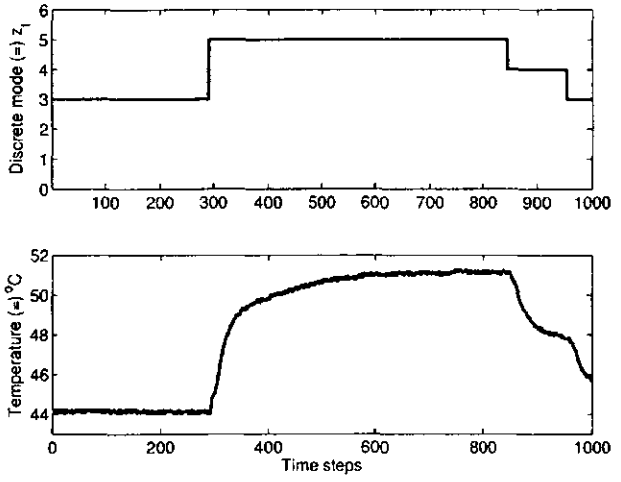

Figure 7.11: Discrete modes and transient response of the heat exchanger. The upper graph shows the different true discrete modes (operating conditions) of the process. The lower graph shows the output water temperature for the different operating conditions. The sampling rate was 2 sec per time step

Figure 7.12 shows a representative example of the tracking performance of the three algorithms when a discrete mode change occurs. By this stage, *PF* has lost track entirely, and *RBPF* fails to recover when the change occurs. *la-RBPF*, on the other hand, recovers reasonably quickly.

7.4 Conclusions

7.4.1 Contributions

- Our main contribution is the *look-ahead Rao-Blackwellized Particle Filtering* algorithm, which is an efficient inference algorithm for the estimation/diagnosis task. *la-RBPF* is a Particle Filtering variant which can be used in real-time applications for technical processes. Practical problems in the control engineering community and others fields can be tackled efficiently using *la-RBPF*.

- The most important *la-RBPF* features are:

  - Low diagnosis error: One-step look-ahead selects good sampling regions, but extra processing must be performed per particle in order to decrease the diagnosis error.

  - Less variance: Based on the Rao-Blackwell theorem, statistically it can be proven that the diagnosis error results have lower variance than standard Particle Filtering algorithms.

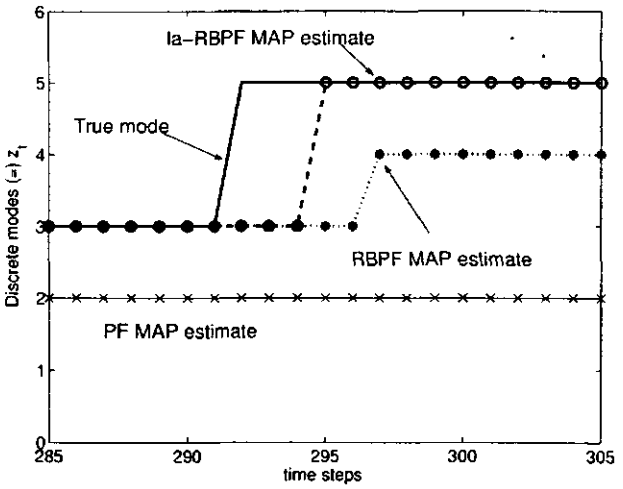

Figure 7.12: Heat exchanger. Tracking when a discrete mode changes. When the discrete mode change occurs, *PF* has lost track entirely, *RBPF* fails to recover, but *la-RBPF* recovers quickly (three time steps) .

  - Low prior probabilities: *la-RBPF* is sampling from the true posterior distribution; it can follow the evidence every time step and detect low probability discrete modes.

- *la-RBPF* has great potential due to the ever-increasing performance of computing hardware and software.
- The jump Markov linear Gaussian *JMLG* model is a special case of a hybrid graphical model in which observable time series data are modelled in terms of unobservable discrete and continuous variables. The proposed learning algorithm, which combines traditional optimization tools such as Least Squares Estimation and the Expectation-Maximization method, showed good results for inference purposes in real technical processes. This is an additional contribution of this research.

7.4.2 RIACS/NASA Ames Research Center applications

The following results are specially considered because allow us to validate the la-RBPF's features in different domains, but mainly for external researches[4]

Testing *la-RBPF* with K-9 planetary rover

The suspension system of the *K-9* planetary rover (described in section 7.2) has a non-linear behaviour; however, under some considerations [de Freitas *et al.*, 2003], it could be modelled as a linear system and *la-RBPF* can be applied for real-time diagnosis. [de Freitas *et al.*, 2003] showed with synthetic and real data that *la-RBPF* presented excellent results (see Appendix F, section F.1.1 for some results with real data.)

Testing *la-RBPF* with Marskohod planetary rover

Marsokhod is medium-sized planetary rover built on a Russian chassis that has been used in field tests from 1993-99 in Russia, Hawaii, and deserts of Arizona and California, [Dearden and Clancy, 2001; Washington, 2000]. The rover

[4] We specially thank Frank Hutter for his observations and recommendations about the *la-RBPF* code.

has six independently driven wheels. Some results with real data presented in [de Freitas *et al.*, 2003] are shown in Appendix F, section F.1.2.

7.4.3 Limitations

- *Manual learning algorithm.*
  The proposed learning algorithm is strongly based on good process knowledge and experience; basically the designer must identify and model all the possible discrete modes and their combinations. During this heavy task the designer isolates the faulty conditions and classifies the different operating conditions; *la-RBPF* do the diagnosis/estimation based on the *JMLG* model and the observations.

- *JMLG model maintenance.*
  La-RBPF generates accurate diagnosis/estimation with the *JMLG* model while the faulty states or operating conditions do not change. If a new faulty state appears or operating conditions change, *la-RBPF* can continue detecting based on the observations, but it is impossible to isolate/classify the new fault state/condition. We have to recompute the model to accommodate new states.

  For the technical process domain we can visualize the possible operating conditions or faulty states. However, the mobile robot domain has the added burden of a changing environment. Currently, learning each model involves a lot of manual work; automating any part of this process would be of great benefit. A robot could periodically enter certain states *voluntarily* in order to update the model parameters for those states. This could be useful as parts age or are replaced.

  There are recursive versions of the Least Squares Estimate (*LSE*) and the Expectation-Maximization (*EM*) methods. An on-line version would be useful. Otherwise, a one-step algorithm (*LSE/EM*) to find all the matrices for a given discrete mode would be very helpful.

- *Number of discrete modes n_z.*
  Normally, industrial processes are operated in a distributed fashion, so diagnosis and estimation tasks have to be done in the same way. Small sections are analyzed and then the results are integrated. However, some processes (e.g. mobile robots) have a large number of discrete modes n_z and they have to be analyzed as a whole. *la-RBPF* can only work off line in this context. Parallel computing or faster software could be a possible solution to this problem, however this option has to be evaluated in practical and economic terms.

- *Linear equations.*
  The Kalman filter is used to compute the continuous states so that we can represent them efficiently. However, the Kalman filter is restricted to systems in which the state equations are linear. We can almost always break up non-linear processes into linear models, but the number of discrete modes n_z grows and the dimension of the problem can become intractable (the previous limitation).

- *Gaussian noise.*
  The Kalman filter is also restricted to Gaussian noise.

- *High dimension in continuous state spaces.*
  The number of particles required for good representations (approximations) of distributions grows exponentially with the dimensionality of the continuous states n_x.

- *Markovian property.*
  The discrete mode variable z_t must follow the Markovian property.

- *Autonomous transition matrix.*
  The proposed *JMLG* model does not permit autonomous transition probabilities. Mode transitions that depend on the continuous behaviour of the system are called autonomous. The probability of each mode transition changes dynamically based on the continuous behaviour of the system and has to be recomputed at every time step.

- *Type of faults.*
  The state space representation in the *JMLG* model considers only permanent abrupt faults. We do not consider incipient or intermittent faults. Indeed, the proposed *la-RBPF* algorithm is restricted to diagnosis/estimation of conditions that can be modelled by the *JMLG* model. This model is powerful but cannot cover all cases.

- *Simultaneous faults.*
  Each combination of faults is considered through a new discrete mode. As the number of possible faults grows, the number of simultaneous faults grows exponentially. However, heuristic information in some domains could be used in order to reduce the number of discrete modes.

- *Operating conditions*
  Same comments that we did for type of faults.

7.5 Future Work

It is our hope that this research demonstrates the usefulness of Particle Filtering algorithms for fault diagnosis using the jump Markov linear Gaussian model. Nevertheless, there is still room for much work to be done. In addition to solving some of the above-mentioned limitations, there are other interesting potential future directions, such as:

7.5.1 Particle Filtering and Qualitative Reasoning

look-ahead RBPF only uses information from the probability distribution and a limited number of observations over time. However, humans can provide very important qualitative information about the state of the system. Qualitative Reasoning seems like the right technique to combine these approaches.

7.5.2 Types of faults

Our experiments concentrated on diagnosing abrupt faults. However, drift faults, which occur slowly over long periods of time, are not detectable within the time periods used in this research. An important issue is the combined problem in which abrupt and drift faults coexist.

7.5.3 JMLG learning procedure

The parameter estimation is a time-consuming process. Integrating some steps and implementing recursive algorithms could greatly improve things, especially when processes change continuously due to plant demands.

7.5.4 Control system

We are not interested in fault diagnosis by itself, but rather as a part of a control system. We presented a probabilistic approach to state estimation and control [Morales-Menéndez *et al.*, 2003] in which *la-RBPF* estimates were used to drive an automatic control system. The experimental domain was the heat exchanger presented in Chapter 5 section 5.4. Preliminary simulated results look great; see Figure 7.13. However, there are many issues to be researched, such as robustness, adaptability and stability, as well as the physical implementation.

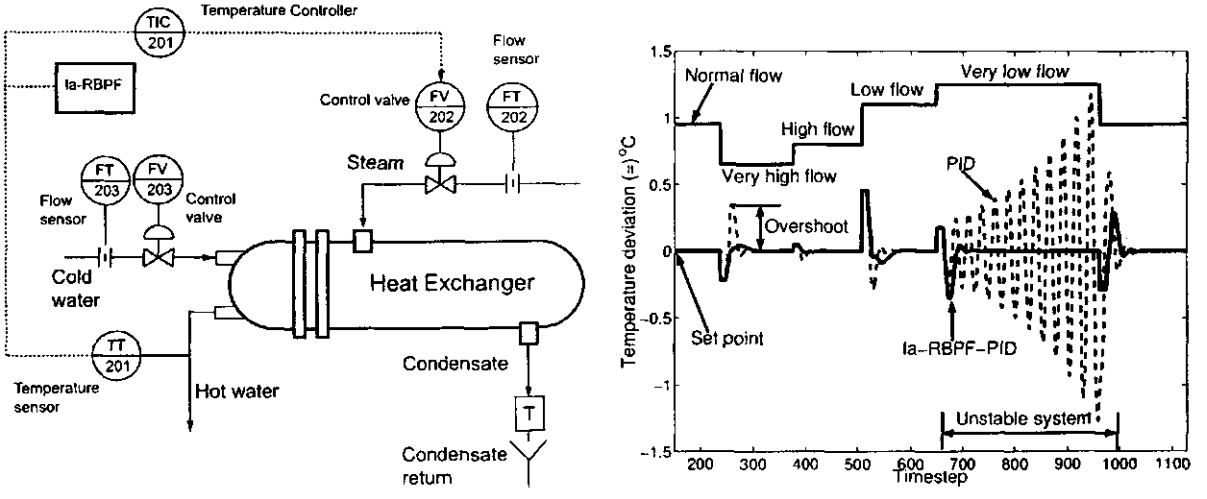

Figure 7.13: Temperature feedback control system. The left plot is a conceptual diagram of a heat exchanger. A feedback control system is shown, where TIC201 represents a PID controller. This controller regulates the output water temperature around a set-point by manipulating the steam flow. The right plot shows the discrete modes of the process over time and the transient responses using the standard PID control system and the improved *la-RBPF-PID* strategy. *la-RBPF-PID* provides a better transient response, less overshoot, and shorter settling time. In contrast, the standard PID control system became unstable in some discrete modes.

7.5.5 Factored Particle Filtering

This follows the ideas of [Ng *et al.*, 2002] and David Poole. When the number of discrete modes grows, *la-RBPF* is restricted to off-line applications. However, it is possible [Ng *et al.*, 2002] to represent the belief state in the form of sets of factored particles, essentially a divide-and-conquer strategy.

7.5.6 Unknown discrete modes

PF, *RBPF* and *la-RBPF* work with a *JMLG* model in which the number of discrete modes is specified and remains constant. However, processes change over time. Research into coping with this situation could be very useful. Some advances are shown in [Hofbaur and Williams, 2002].

7.5.7 Hybrid observer

For the jump Markov linear Gaussian (*JMLG*) model, it is sometimes difficult or impractical to generate faults and build dynamic models during these faulty conditions. However, we can model the process in different normal operating conditions which characterize the full process performance.

In this context, one could use the *la-RBPF* inference algorithm as a typical hybrid observer, Figure 7.14. This hybrid observer could be part of a standard Fault Detection and Isolation (*FDI*) system. *la-RBPF* can compute the most likely discrete modes $\widehat{z}_t$, continuous states $\widehat{x}_t$ and expected observations $\widehat{y}_t$ based on the *JMLG* model. A residual signal can be generated, $\varepsilon_t = y_t - \widehat{y}_t$. The *FDI* system would work by considering both the residual signal, ε_t, and the estimated modes $\widehat{z}_t$. Using both residuals and modes, the *FDI* system would validate among hybrid states, i.e. residual behaviour (exceeding some limit) after a mode transition does not necessarily correspond to a fault.

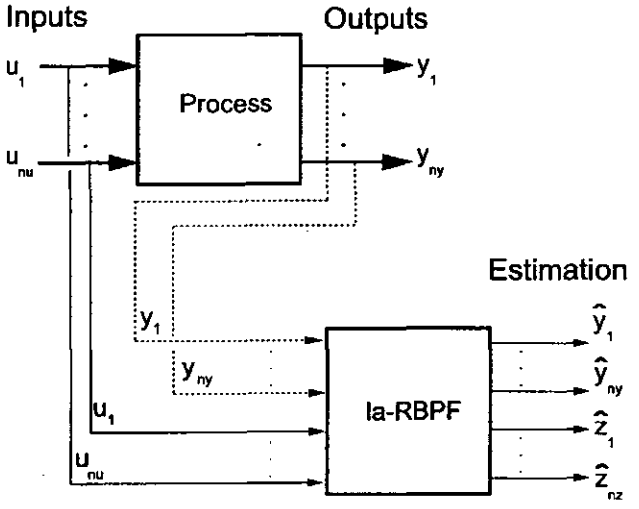

Figure 7.14: Look-ahead RBPF as a hybrid observer. Having both the residual signal $\varepsilon_t = y_t - \widehat{y}_t$ and estimated modes $\widehat{z}_t$ allows better *FDI* performance.

Bibliography

[Akashi and Kumamoto, 1977] H Akashi and H Kumamoto. Random sampling approach to state estimation in switching environments. *Automatica*, 13:429–434, 1977.

[Antonelli, 2003] G Antonelli. *Underwater robots. Motion and force control of vehicle-manipulator systems.* Springer, Hidelberg D, 2003.

[Arroyo-Figueroa and Sucar, 1999] G Arroyo-Figueroa and L E Sucar. A temporal Bayesian network for diagnosis and prediction. In K Blackmond and H Prade, editors, 15^{th} *Conference on Uncertainty in Artificial Intelligence*, pages 13–20, Stockholm, Sweden, 1999. Morgan Kaufmann Publishers, Inc.

[Arroyo-Figueroa *et al.*, 1998] G Arroyo-Figueroa, L E Sucar, and A Villavicencio. Probabilistic temporal reasoning and its application to fossil power plant. *Expert Systems with Applications*, 15:317–324, 1998.

[Arroyo-Figueroa *et al.*, 2000] G Arroyo-Figueroa, Y Alvarez, and L E Sucar. SEDRET – an intelligent system for the diagnosis and prediction of events in power plants. *Expert Systems with Applications*, 18:75–86, 2000.

[Bar-Shalom and Li, 1993] Y Bar-Shalom and X R Li. *Estimation and Tracking: Principles, Techniques and Software.* Artech House, Boston, 1993.

[Bar-Shalom and Li, 1995] Y Bar-Shalom and X R Li. *Multitarget-Multisensor Tracking: Principles and Techniques.* YBS, 1995.

[Basseville and Nikiforov, 1993] M Basseville and I Nikiforov. *Detection of abrupt changes: theory and applications.* Prentice Halls, Englewood Cliffs, NJ, 1993.

[Baum *et al.*, 1970] L E Baum, T Petrie, G Soules, and N Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171, 1970.

[Bengio and Frasconi, 1995] Y Bengio and P Frasconi. An input-output hmm architecture. In G Tesauro, D S Touretzky, and T K Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 427–434. MIT Press, Cambridge, MA, 1995.

[Berleant and Kuipers, 1992] D Berleant and B Kuipers. Qualitative-numeric simulation with q3. In B Faltings and P Strass. MIT Press, editors, *Recent advances in qualitative physics*, pages 3–16, Cambridge, MA, 1992.

[Berleant and Kuipers, 1997] J D Berleant and B Kuipers. Qualitative amd quantitative simulation: Bridging the gap. *Artificial Intelligence*, 2(95):215–255, 1997.

[Berleant, 1991] J D Berleant. *The use of Partial Quantitative Information with Qualitative Reasoning*. PhD thesis, University of Texas at Austin, Austin, Tx, USA, 1991.

[Bernardo and Smith, 1994] J M Bernardo and A F M Smith. *Bayesian Theory*. Wiley Series in Applied Probability and Statistics, 1994.

[Boskŏivć and Mehra, 2002] J D Boskŏivć and R K Mehra. Multiple model-based adaptive reconfigurable formation flight control design. In 41^{st} *IEEE Conference on Decision and Control*, pages 1263–1268, Las Vegas, Nevada USA, 2002.

[Boyen and Koller, 1998] X Boyen and D Koller. Tractable inference of complex stochastic processes. In 11^{th} *Conference on Uncertainty in Artificial Intelligence*, pages 33–42, Morgan Kaufmann, 1998.

[Caccavale, 1998] F Caccavale. Experiments of observer-based fault detection for an industrial robot. In *IEEE International Conference on control applications*, Trieste, Italy, September 1998.

[Casella and Berger, 1990] G Casella and R L Berger. *Statistical Inference*. Brooks Cole, Pacific Grove, CA, 1990.

[Casella and Robert, 1996] G Casella and C P Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[Chang and Athans, 1978] C B Chang and M Athans. State estimation for discrete systems with switching parameters. *IEEE Trans on Aerospace and Electronic Systems*, AES-14(3):418–424, 1978.

[Chen and Howell, 2001] J Chen and J Howell. A self-validating control system based approach to plant fault detection and diagnosis. *Computers and Chemical Engineering*, 25:337–358, 2001.

[Chen *et al.*, 2001] Y Chen, S Shi, and Z Weng. A novel robust fault identification approach for a class of discrete systems. In *IECON'01 The* 27^{th} *Annual Conference of the IEEE Industrial Electronics Society*, pages 704–706, Denver, Co, USA, 2001.

[Clancy and Kuipers, 1998] D J Clancy and B Kuipers. Qualitative simulation as a temporally-extended constraint satisfaction problem. In 15^{th} *National Conference on Artificial Intelligence (AAAI-98)*. AAAI/MIT Press, 1998.

[Clancy, 1997] D J Clancy. *Solving complexity and ambiguity problems within Qualitative Simulation*. PhD thesis, University of Texas at Austin, Austin, Tx, USA, 1997.

[Clancy, 1998] D J Clancy. Qualitative, model-based diagnosis of complex physical devices. *IEEE International Conference on Systems, Man, and Cybernetics*, 3:3012–3019, 1998.

[Cover and Thomas, 1991] T M Cover and J A Thomas. *Elements of information theory*. Wiley Series in Telecommunications, New York, 1991.

[Crisan and Doucet, 2000] D Crisan and A Doucet. Convergence of sequential Monte Carlo methods. Technical Report CUED/F-INFENG/TR 381, Cambridge University Engineering Department, June 2000.

[Crisan and Doucet, 2002] D Crisan and A Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746, 2002.

[Crisan *et al.*, 1999] D Crisan, P Del Moral, and T Lyons. Discrete filtering using branching and interacting particle systems. *Markov Processes and Related Fields*, 5(3):293–318, 1999.

[Crisan, 2001] D Crisan. Particle filters - a theoretical perspective. In A Doucet, N de Freitas, and N J Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[Dash *et al.*, 2001] S Dash, R Rengaswamy, and V Venkatasubramanian. A novel interval-halving algorithm for process trend identification. In *Preprints of 4^{th} IFAC Workshop on On-line Fault Detection and Supervision in the Chemical Process Industries*, pages 173–178, Jejudo Island, Korea, 2001.

[Dash *et al.*, 2003] S Dash, R Rengaswamy, and V Venkatasubramanian. Fuzy-logic based trend classification for fault diagnosis of chemical processes. *Computers and Chemical Engineering*, 27:347–362, 2003.

[Davis, 1984] R Davis. Diagnostic reasoning based on structure and behavior. *Artificial intelligence*, 24:347–410, 1984.

[de Freitas *et al.*, 2003] N de Freitas, R Dearden, F Hutter, R Morales-Menéndez, J Mutch, and D Poole. Diagnosis by a waiter and a Mars explorer. *submitted to IEEE, special issue on Sequential State Estimation*, 2003.

[de Freitas, 2001] N de Freitas. Rao-Blackwellised particle filtering for fault diagnosis. In *IEEE Aerospace Conference*, 2001.

[de Kleer and Brown, 1984] J de Kleer and J Brown. A qualitative physics based on confluences. *Artificial intelligence*, 24:7–83, 1984.

[de Kleer and Williams, 1987] J de Kleer and B Williams. Diagnosing multiple faults. *Artificial intelligence*, 32:97–130, 1987.

[de Silva *et al.*, 1992] V de Silva, G de Souza, and G Zaverucha. An integration of neural networks and nonmonotonic reasoning for power systems diagnosis. *International Conference on neural networks*, 3:1409–1413, 1992.

[Dean and Kanazawa, 1989] T Dean and K Kanazawa. A model for reasoing about persistence and causation. *Computational intelligence*, 5(3):142–150, 1989.

[Dearden and Clancy, 2001] R Dearden and D Clancy. Particle filters for real-time fault detection in planetary rovers. In *International Workshop on Diagnosis (DX)*, 2001.

[Dempster *et al.*, 1977] A P Dempster, N M Laird, and D B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38, 1977.

[Dinca *et al.*, 1999] L Dinca, T Aldemir, and G Rizzoni. Fault detection and identification in dynamic systems with noisy data and parameter/modeling uncertainties. *Reliability Engineering and System Safety*, 65:17–28, 1999.

[Ding and Frank, 1999] X Ding and P Frank. On-line fault detection in uncertain systems using adaptive observers. *European J. of diagnosis and safety in automation*, 3:9–21, 1999.

[Ding *et al.*, 1990] X Ding, P Frank, and L Guo. Fault detection via adaptive observers based on orthogonal functions. In *IFAC symposium on advanced information processing in automatic control*, pages 95–100, 1990.

[Ding *et al.*, 1994] X Ding, L Guo, and P Frank. Parametrization of linear observers and its application to observer design. *IEEE Trans Automatic Control*, pages 1648–1652, 1994.

[Doucet and Gordon, 1999] A Doucet and N J Gordon. Simulation-based optimal filter for manoeuvring target tracking. In *SPIE Signal and Data Processing of Small Targets*, volume SPIE 3809, 1999.

[Doucet *et al.*, 1999] A Doucet, N J Gordon, and V Krishnamurthy. Particle filters for state estimation of jump Markov linear systems. Technical Report CUED/F-INFENG/TR 359, Cambridge University Engineering Department, Sep 1999.

[Doucet *et al.*, 2000a] A Doucet, N de Freitas, K Murphy, and S Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In C Boutilier and M Godszmidt, editors, *Uncertainty in Artificial intelligence*, pages 176–183. Morgan Kaufmann Publishers, 2000.

[Doucet *et al.*, 2000b] A Doucet, S Godsill, and C Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[Doucet *et al.*, 2001] A Doucet, N de Freitas, and N J Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[Doucet, 1997] A Doucet. *Monte Carlo Methods for Bayesian Estimation of Hidden Markov Models. Application to Radiation Signals*. PhD thesis, University Paris-Sud, Orsay, France, 1997. Chapters 4 and 5 in English.

[Doucet, 1998] A Doucet. On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR 310, Dept of Engineering, Cambridge University, 1998.

[Dukelow, 1991] S G Dukelow. *The Control of Boilers*. Instrument Society of America, 1991.

[Duyar and Eldem, 1992] A Duyar and V Eldem. Fault detection and diagnosis in propulsion systems: a real identification approach. In *IFAC/IMACS symposium SAFEPROCESS'91*, pages 473–478, 1992.

[Dvorak and Kuipers, 1989] D Dvorak and B Kuipers. Model-based monitoring of dynamic systems. In 11^{th} *IJCAI*, pages 1238–1243, Detroit, USA, August 1989.

[Dvorak, 1992] D L Dvorak. *Monitoring and Diagnosis of Continuous Dynamic Systems usign Semiquantitative simulation*. PhD thesis, University of Texas at Austin, Austin, Tx, USA, 1992.

[Elinas *et al.*, 2002] P Elinas, J Hoey, D Lahey, D Montgomery, D Murray, S Se, and J Little. Waiting with josé, a vision-based mobile robot. In *International Conference on Robotics and Automation*, 2002.

[Elliot *et al.*, 1995] R J Elliot, L Aggoun, and J B Moore. *Hidden Markov Models: estimation and control*. Springer-Verlag, New York, 1995.

[Elliott, 1994] R J Elliott. Exact adaptive filters for Markov chains observed in Gaussian noise. *Automatica*, 30(9):1399–1408, 1994.

[Enste and Uecker, 2000] U Enste and F Uecker. Use of supervisory information in process control. *Control Engineering Journal*, pages 234–241, October 2000.

[Fan *et al.*, 1993] J Y Fan, M Nikolaou, and R.E. White. An approach to fault diagnosis of chemical process via neural networks. *AIChE Journal*, 39:82–88, 1993.

[Fisher and Maybeck, 2002] K A Fisher and P S Maybeck. Multiple model adaptive estimation with filter spawning. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):755–768, 2002.

[Fisher, 1999] K A Fisher. Multiple model adaptive estimation using filter spawning. Master's thesis, School of Engineering, Air Force Institute of Technology, Wrigth-Patterson ADB, OH, 1999.

[Forbus, 1984] K Forbus. Qualitative process theory. *Artificial intelligence*, 24:85–168, 1984.

[Forbus, 1986] K Forbus. Interpreting measurements of physical systems. In 5^{th} *National Conference on Artificial intelligence*, pages 113–117, 1986.

[Frank and Ding, 1997] P Frank and X Ding. Survey of robust residual generation and evaluation methods in observer-based fault detection systems. *J. Proc. Cont*, 7(6):403–424, 1997.

[Frank and Koppen-Seliger, 1997] P Frank and B Koppen-Seliger. Fuzzy logic and neural network: Applications to fault diagnosis. *International Journal of Approximate Reasoning*, 16:67–78, 1997.

[Frank *et al.*, 2000] P Frank, E Alcorta Garcia, and B Koppen-Seliger. Modelling for fault detection and isolation versus modelling for control. *Mathematics and computers in simulation*, 53:259—271, 2000.

[Frank, 1990] P Frank. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy – a survey and some new results. *Automatica*, 26:459–474, 1990.

[Frank, 1992] P Frank. Robust model-based fault detection in dynamic systems. *Preprints of IFAC symposium : on-line fault detection in the chemical process industries*, pages 1–13, 1992.

[Fraser and Dimitriadis, 1993] A M Fraser and A Dimitriadis. Forecasting probability densities by using hidden Markov models with mixted states. In A S Wiegand and N A Gershenfeld, editors, *Time series prediction: Forecasting the future and understanding the past, SFI studies in the Sciences of Complexity*, volume XV, pages 265–282. Addsion-Wesley, Reading MA, 1993.

[Garza *et al.*, 2001] L Garza, F Cantú, and S Acevedo. Fault diagnosis in power transmission networks with hybrid diagnosis system. In 14^{th} *IEEE Summer Power Meeting and Industrial Applications*, pages 149–154, Acapulco, Gro, México, 2001.

[Genesereth, 1984] M Genesereth. The use of design descriptions in automated diagnosis. *Artificial intelligence*, 24:411–436, 1984.

[Gertler, 1997] J Gertler. Fault detection and isolation using parity relations. *Control Eng. Practice*, 5(5):653–661, 1997.

[Gertler, 1998] J Gertler. *Fault detection and diagnosis in engineering systems*. Marcel Dekker, Inc., 1998.

[Ghahramani and Hinton, 1996] Z Ghahramani and G E Hinton. Parameter estimation for linear dynamical system. Technical Report CRG-TR-96-2, Department of Computer Science, University of Toronto, Toronto, 1996.

[Ghahramani and Hinton, 1998] Z Ghahramani and G E Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):963–996, 1998.

[Goel *et al.*, 1999] P Goel, S Roumeliotis, and G Sukhatme. Robust localization using relative and absolute position estimates. In *IEEE/RSJ International Conference on intelligent robots and systems*, October 1999.

[Goel *et al.*, April 2000] P Goel, G Dedeoglu, S Roumeliotis, and G Sukhatme. Fault detection and identification in a mobile robot using multiple model estimation and neural networks. In *IEEE International Conference on robotics and automation*, pages 2302–2309, San Francisco, CA, USA, April, 2000.

[Gordon *et al.*, 1993] N Gordon, D Salmond, and Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proceedings-F*, 140(2):107–113, April 1993.

[Hamilton, 1994] J D Hamilton. *Time series analysis*. Princeton University Press, Princeton, NJ, 1994.

[Hanlon and Maybeck, 2000] P D Hanlon and P S Maybeck. Multiple-model adaptive estimation using a residual correlation kalman filter bank. *IEEE Transactions on Aerospace and Electronic Systems*, 36(2):393–406, 2000.

[Hashimoto *et al.*, Oct 29 Nov 03 2001] M Hashimoto, H Kawashima, T Nakagami, and F Oba. Sensor fault detection and identification in dead-reckoning system of mobile robot: interacting multiple model approach. In *IEEE/RSJ International Conference on intelligent robots and systems*, pages 1321–1326, Maui, Hawaii, USA, Oct 29 - Nov 03, 2001.

[Higuchi, 1997] T Higuchi. Monte Carlo filter using the genetic algorithm operators. *Journal of Statistical Computation and Simulation*, 59(1):1–23, 1997.

[Himmelblau, 1978] D Himmelblau. *Fault detection and diagnosis in chemical and pretrochemical processes*. Elsevier Scientific Publishing Company, USA, 1978.

[Hofbaur and Williams, 2002] M W Hofbaur and B C Williams. Hybrid diagnosis with unkown behaviour modes. In 13^{th} *International wrokshop on Principles of Diagnosis*, pages 97–105, Semmering, Austria, 2002.

[Hutter and Dearden, 2003a] F Hutter and R Dearden. Efficient on-line fault diagnosis for non-linear systems. In 7^{th} *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2003.

[Hutter and Dearden, 2003b] F Hutter and R Dearden. The Gaussian particle filter for diagnosis of non-linear systems. In 14^{th} *International Workshop on Principles of Diagnosis*, Washington, DC, 2003.

[Isard and Blake, 1996] M Isard and A Blake. Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision*, pages 343–356, Cambridge, UK, 1996.

[Isard and Blake, 1998] M Isard and A Blake. Condensation: conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998.

[Isermann, 1997a] R Isermann. Supervision, fault-detection and fault-diagnosis methods - an introduction. *Control engineering practice*, 5(5):639–652, 1997.

[Isermann, 1997b] R Isermann. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control engineering practice*, 5(5):709–719, 1997.

[Jordan *et al.*, 1999] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[Julier and Uhlmann, 1997] S Julier and J Uhlmann. A new extension of Kalman filter to nonlinear systems. In *AeroSense : the* 11^{th} *International symposium on aerospace/defense sensing, simulationn and controls*, 1997.

[Kalman and Bucy, 1961] R Kalman and R Bucy. New results in linear filtering and prediction theory. *Transactions of the ASME (Jounal of basic engineering)*, 83D:95–108, 1961.

[Kalman, 1960] R Kalman. A new approach to linear filtering and prediction problems. *Basic Engineering*, 82:34–45, 1960.

[Kanazawa *et al.*, 1995] K Kanazawa, D Koller, and S Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In 11^{th} *Conference on uncertainty in Artificial intelligence*, pages 346–351, Morgan Kaufmann, 1995.

[Kay and Ungar, 2000] H Kay and L H Ungar. Estimating monotonic functions and their bounds. *AIChE*, 46(12):2426–2434, 2000.

[Kay *et al.*, 2000] H Kay, B Rinner, and B Kuipers. Semi-quantitative system identification. *Artificial Intelligence*, 119:103–140, 2000.

[Keller, 1999] J Y Keller. Fault isolation filter design for linear stochastic systems. *Automatica*, 25:1701–1706, 1999.

[Kim, 1994] C J Kim. Dynamic linear models with Markov-switching. *Journal of Econometrics*, 60:1–22, 1994.

[Kitagawa, 1996] G Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5:1–25, 1996.

[Kiupel and Frank, 1993] N Kiupel and P Frank. Process supervision with the aid of fuzzy logic. *IEEE SMC conference*, 1993.

[Koller and Lerner, 2001] D Koller and U Lerner. Sampling in factored dynamic systems. In A Doucet, N de Freitas, and N J Gordon, editors, *Sequential Monte Carlo Methods in Practice*, pages 470–490. Springer-Verlag, 2001.

[Kong *et al.*, 1994] A Kong, J S Liu, and W H Wong. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.

[Koppen-Seliger and Frank, 1995] B Koppen-Seliger and P Frank. Fault detection and isolation in technical processes with neural networks. In 34^{th} *Conference on decision and control CDC*, New Orleans, 1995.

[Koutsoukos *et al.*, 2002] X Koutsoukos, J Kurien, and F Zhao. Monitoring and diagnosis of hybrid systems using particle filtering methods. In 15^{th} *International Symposium on Mathematical Theory of networks and Systems*, Notre Dame, IN, 2002.

[Krishnamurthy and Moore, 1993] V Krishnamurthy and J B Moore. On-line estimation of hidden Markov model parameters based on the Kullback-Leibler information measure. *IEEE Transactions on Signal Processing*, 41(8):2557–2573, Aug 1993.

[Kuipers, 1984] B Kuipers. Common sense reasoning about causality: deriving behavior from structure. *Artificial intelligence*, 24:169–204, 1984.

[Kuipers, 1986] B Kuipers. Qualitative simulation. *Artificial intelligence*, 29:289–338, 1986.

[Kuipers, 1987] B Kuipers. Qualitative simulation as causal explanation. *IEEE Transactions on systems, man and cybernetics*, 3:432–444, 1987. SMC-17.

[Kuipers, 1994] B Kuipers. *Qualitative Reasoning : Modeling and Simulation with Incomplete Knowledge*. MIT Press, Cambridge, MA, 1994.

[Kuipers, 2001] B Kuipers. Qualitative simulation. In R A Meyers, editor, *Encyclopedia of Physical Science and Technology*. Academic Press, 2001.

[Lafortune *et al.*, 2001] S Lafortune, D Teneketzis, R Sengupta, M Sampath, and K Sinnamohideen. Failure diagnosis of dynamic systems: An approach based on discrete event systems. In *American Control Conference*, pages 2058–2071, Arlington, VA, 2001.

[Landau, 1990] I D Landau. *System identification and control design.* Prentice Hall, 1990.

[Leger, 2000] C Leger. *Darwin2K: An evolutionary approach to automated design for robotics.* Kluwer Academic Publishers, 2000.

[Leonhardt and Ayoubi, 1997] S Leonhardt and M Ayoubi. Methods of fault diagnosis. *Control Eng. Practice*, 5(5):683–692, 1997.

[Lerner *et al.*, 2000] U Lerner, R Parr, D Koller, and G Biswas. Bayesian fault detection and diagnosis in dynamic systems. In 17^{th} *National Conference on Artificial Intelligence*, 2000.

[Lerner *et al.*, 2002] U Lerner, B Moses, M Scott, S McIlraith, and D Koller. Monitoring a complex physical system using a hybrid dynamic Bayes net. In 18^{th} *Annual Conference on Uncertainty in Artificial Intelligence*, Edmonton, Canada, 2002.

[Liu and Chen, 1998] J Liu and R Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93:1032–1044, 1998.

[Liu and Si, 1997] B Liu and J Si. Fault isolation filter design for time-invariant systems. *IEEE Transactions on Automatic Control*, 21:704–707, 1997.

[Ljung, 1987] L Ljung. *System Identification: Theory for the User.* Prentice-Hall, 1987.

[Logothetis and Krishnamurthy, 1999] A Logothetis and V Krishnamurthy. Expectation-maximization algorithms for map estimation of jump Markov linear systems. *IEEE Transactions Singal Processing*, 47:2139–2156, 1999.

[Luyben, 1989] W L Luyben. *Process Modeling, Simulation and Control for Chemical Engineers.* Mc Graw-Hill Publishing Company, second edition, 1989.

[Magill, 1985] D Magill. Optimal adaptive estimation of sampled stochastic processes. *IEEE Transactions on automatic control*, 10(4):434–439, 1985.

[Mazor *et al.*, 1998] E Mazor, A Averbuck Y Bar-Shalom, and J Dagan. Interacting multiple model methods in target tracking. A Survey. *IEEE Transactions Aerospace Electron. Syst.*, 34:103–123, 1998.

[McIlraith *et al.*, 1999] S McIlraith, G Biswas, D Clancy, and V Gupta. Hybrid systems diagnosis. In 10^{th} *International workshop on principles of diagnosis*, pages 193–203, Loch Awe, Scotland, June 1999.

[McIlraith, 2000] S McIlraith. Diagnosing hybrid systems : a Bayesian model selection approach. In 11^{th} *International workshop on principles of diagnosis*, pages 140–146, Morelia, Mexico, June 2000.

[Medvedev, 1995] A Medvedev. Fault detection and isolation by a continuous parity space method. *Automatica*, 31(7):1039–1044, 1995.

[Molle and Edgar, 1990] D Dalle Molle and T F Edgar. Qualitative modeling of chemical reaction system. In Mavrovouniotis M.L, editor, *Artificial Intelligence in Process Engineering*, pages 1–36. Academic Press, Inc, 1990.

[Molle, 1989] D Dalle Molle. Qualitative simulation of dynamic chemical process. Technical Report AI89-107, University of Texas at Austin, Austin, Tx, USA, 1989.

[Monahan, 1982] G Monahan. A survery of partially observable Markov decision processes: Theory, models and algorithms. *Management Science*, 1982.

[Montmain and Gentil, 1993] J Montmain and S Gentil. Decision-making in fault detection: a fuzzy approach. In *TOOLDIAG'93 International Conference on fault diagnosis*, Toulouse, France, 1993.

[Morales *et al.*, 2001] D Morales, F Valles, and R Ramirez. Fault detection and isolation FDI techniques for the design of an supervisory system of an industrial thermal process. In *ISA Technical Conference*, pages 223–228, Houston, Tx, 2001.

[Morales-Menéndez *et al.*, 2002] R Morales-Menéndez, N de Freitas, and D Poole. Real-time monitoring of complex industrial processes with particle filters. In *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2002. MIT Press.

[Morales-Menéndez *et al.*, 2003] R Morales-Menéndez, N de Freitas, and D Poole. Estimation and control of industrial processes with particle filters. In *American Control Conference*, 2003.

[Morales-Menéndez, 1992] R Morales-Menéndez. Process identification and digital controller design. Master's thesis, ITESM campus Monterrey, México, July 1992.

[Mosterman and Biswas, 1999] P Mosterman and G Biswas. Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on systems, man, and cybernetics*, 29(6):554–565, 1999. Part A.

[NASA, 1999] NASA. Mars Climate Orbiter mishap investigation board Phase I Report. Technical report, NASA, Nov 10 1999.

[NASA, 2000] NASA. Mars program independent assessment team Summary Report. Technical report, NASA, March 14 2000.

[Ng *et al.*, 2002] B Ng, L Peshkin, and A Pfeffer. Factored particles for scalable monitoring. In 18^{th} *Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 2002.

[Ng, 1990] H Ng. Model-based, multiple fault diagnosis of time-varying continuous physical devices. In 6^{th} *Conference on Artificial intelligence. applications*, Santa Barbara, CA, 1990.

[Ogata, 1995] K Ogata. *Discrete-Time Control Systems*. Prentice Hall, second edition, 1995.

[Patton and Chen, 1991] R Patton and J Chen. Parity space approach to model-based fault diagnosis. a tutorial survey and some results. *IFAC SAFEPROCESS Symposium Baden-Baden*, 1:239–255, 1991.

[Pitt and Shephard, 1999] M Pitt and N Shephard. Filtering via simulation: auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999.

[Polycarpou and Vemuri, 1995] M Polycarpou and A Vemuri. Learning methodology for failure detection and accomodation. *IEEE Control systems intellegence and learning*, 15(3):16–24, 1995.

[Poole *et al.*, 1998] D Poole, A Mackworth, and R Goebel. *Computational Intelligence – A logical Approach*. Oxford University Press, New York, 1998.

[Poole, 1987] D Poole. A logical framework to default reasoning. *Artificial intelligence*, 36(1):27–47, 1987.

[Poole, 1989] D Poole. Normality and faults in logic-based diagnosis. In 11^{th} *International Joint Conference on Artificial intelligence*, volume 2, pages 1304–1310, Detroit, 1989.

[Poole, 1993] D Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial intelligence*, 64(1):81–129, 1993.

[Poole, 1997] D Poole. The independent choice logic for modeling multiple agents under uncertainty. *Artificial intelligence*, 94(1-2):7–56, 1997.

[Poole, 2000] D Poole. Abducing through negation as failure: Stable models within the independent choicer logic. *Journal of logic programming*, 44:5–35, 2000.

[Rauch *et al.*, 1965] H E Rauch, F Tung, and C T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, August 1965.

[Reiter, 1987] R Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32:57–95, 1987.

[Rinner and Kuipers, 1999a] B Rinner and B Kuipers. Monitoring piecewise continuous behavior by refining trackers and models. *Hybrid systems and Artificial intelligence: modeling, analysis and control of discrete+continuous systems*, AAAI technical report SS-99-05:164–169, 1999.

[Rinner and Kuipers, 1999b] B Rinner and B Kuipers. Monitoring piecewise continuous behaviors by refining semi-quantitative trackers. In 16^{th} *IJCAI*, pages 1080–1086, Stockholm, Sweden, 1999.

[Ripley, 1987] B D Ripley. *Stochastic Simulation*. Wiley, New York, 1987.

[Roumeliotis *et al.*, May 1998] S Roumeliotis, G Sukhatme, and G Bekey. Fault detection and identification in a mobile robot using multiple-model estimation. In *IEEE International Conference on robotics and automation*, pages 2223–2228, Leuven, Belgium, May, 1998.

[Rudas, 1991] I Rudas. Fault detection in robots using stochastic filtering. In *IECON*, pages 1147–1152, 1991.

[Rumelhart *et al.*, 1986] D E Rumelhart, G E Hinton, and R J Williams. Learning internal representations by error propagation. In D E Rumelhart and J L McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pages 318–362, Cambridge, MA, 1986.

[Sampath, 2001] M Sampath. A hybrid approach to failure diagnosis of industrial systems. In *American Control Conference*, pages 2077–2082, Arlington, VA, 2001.

[Schneider, 1993] H Schneider. Implementation of a fuzzy concept for supervision and fault detection of robots. In *EUFIT'93 First European congress on fuzzy and intelligent technologies*, pages 775–780, Aachen, 1993.

[Shumway and Stoffer, 1991] R H Shumway and D S Stoffer. Dynamic linear models with switching. *J of the American Statistical Association*, 86(415):763–769, Sep 1991.

[Smith and Corripio, 1997] C A Smith and A B Corripio. *Principles and Practice of Automatic Process Control*. John Wiley & Sons, second edition, 1997.

[Smith, 1972] C Smith. *Digital Computer Process Control*. Intext Educational Publishers, 1972.

[Staroswiecki and Comtet-Varga, 2001] M Staroswiecki and G Comtet-Varga. Analytical redundancy relations for fault detection and isolation in algebraic dynamic systems. *Automatica*, 37:687–699, 2001.

[Struss, 1997] P Struss. Fundamentals of model-based diagnosis of dynamic systems. In 15^{th} *International join Conference con Artificial intelligence*, pages 480–485, Nagoya, Japan, 1997.

[Subramanian, 1995] S Subramanian. Qualitative multiple-fault diagnosis of continuous dynamic systems using behavioral modes. Technical Report TR C-239, AI Laboratory, University of Texas at Austin, Austion, Tx, USA, 1995.

[Terra and Tinos, 2001] M Terra and R Tinos. Fault detection snf isolation in robotic manipulators via neural networks: A comparison among three architectures of residual analysis. *J. of robotic systems*, 18(7):357–374, 2001.

[Thrun *et al.*, 2002] S Thrun, J Langford, and V Verma. Risk sensitive particle filters. In S Becker T. G Dietterich and Z Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[Thrun, 2002] S Thrun. Particle filters in robotics. In 18^{th} *Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 2002.

[Tugnait, 1982] J K Tugnait. Adaptive estimation and identification for discrete systems with Markov jump parameters. *IEEE Transactions on Automatic Control*, AC-27(5):1054–1065, 1982.

[Valles *et al.*, 1998] F Valles, Y Sanchez, and R Morales-Menéndez. Data acquisition system and control. Technical report, Control Engineering Dept, ITESM campus Monterrey, 1998.

[Valles, 2001] F Valles. *Fault Detection and Isolation techniques for supervision of adaptive control systems*. PhD thesis, Center of Artifial Intelligence, ITESM campus Monterrey, Monterrey, NL, México, 2001.

[van der Merwe *et al.*, 2001] R van der Merwe, A Doucet, N de Freitas, and E Wan. The unscented particle filter. In T K Leen, T G Dietterich, and V Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 584–590, MIT Press, 2001.

[Verma *et al.*, 2001] V Verma, J Langford, and R Simmons. Non-parametric fault identification for space rovers. In *International Symposium on Artificial Intelligence and Robotics in Space (iSAIRAS)*, June 2001.

[Verma *et al.*, 2002] V Verma, J Fernandez, and R Simmons. Probabilistic models for monitoring and fault diagnosis. In R Chatila, editor, *The Second IARP and IEEE/RAS Joint workshop on technical challenges for dependable robots in human environment*, Toulouse, France, October 2002.

[Visinsky *et al.*, 1995] M L Visinsky, J R Cavallaro, and I D Walker. A dynamic fault tolerance framework for remote robots. *IEEE Transactions on Robotics and Automation*, 11:477–490, 1995.

[Vora *et al.*, 1997] N Vora, S S Tambe, and B D Kulkarni. Counterpropagation neural networks for fault detection and diagnosis. *Computers Chemical Engineering*, 21(2):177–185, 1997.

[Washington, 2000] R Washington. On-board real-time state and fault identification for rovers. In *IEEE International Conference on robotics and automation*, 2000.

[Watanabe *et al.*, 1994] K Watanabe, S Hirota, L Hou, and D M Himmelblau. Diagnosis of multiple simultaneous fault via hierarchical artificial neural networks. *AIChE Journal*, 40:839–848, 1994.

[West, 1993] M West. Mixture models, Monte Carlo, Bayesian updating and dynamic models. *Computing Science and Statistics*, 24:325–333, 1993.

[Whiteley and Davis, 1994] J R Whiteley and J F Davis. A similarity-based approach to interpretation of sensor data using adaptive resonance theory. *Computers Chemical Engineering*, 18:637–661, 1994.

[Zadeh, 1978] L Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 1:3–28, 1978.

[Zhu and Backx, 1993] Y Zhu and T Backx. *Identification of multivariable industrial process – for simulation, diagnosis and control.* Springer-Verlag, 1993.

[Zhuang and Frank, 1997] Z Zhuang and P Frank. Qualitative observer and its application to FDI systems. *Proc. Instn. Mech. Eng*, 211(4):253–262, 1997.

Appendix A

Diagnosis in Dynamic Systems

A.1 Definitions

Some important definitions in dynamic systems are:

1. *Static versus dynamic systems.*
   A system is static if its outputs at any time depend only on its inputs acting at the same time. In contrast, the outputs of a dynamic system are affected by present and/or past inputs. The model of a static system consist of *algebraic equations* while dynamic systems are described by *differential* or *difference* equations. Dynamic systems in their steady state can also be characterized by static models.

2. *Linear versus non-linear models.*
   A linear model, static or dynamic, is one to which the *principle of superposition* applies, that is, where the response to combination of input is the same as the sum of the individual responses. This implies that algebraically the model contains only terms which are linear in the variables. Nonlinear models have terms which are nonlinear in the variables, such as square of a variable. A system is linear if it can be exactly characterized by a linear model. While most physical systems are, in fact, non-linear, approximate linear models can usually be derived which are valid in the vicinity of an operating point. We exploit this idea through this research, in order to keep as simple as possible the modelling task.

3. *Continuous-time versus discrete-time models.*
   Continuous-time variables are those which exist (their values are defined) for any value of time. In contrast, *Discrete-time variables* exist (are defined) only for certain time instants. Continuous-time models relate continuous variables to each other while discrete-time models relate discrete-ones. Continuous models are differential equations or other model forms derived from the latter, discrete models are difference equations or their derivatives. Most physical systems are continuous by nature. However, in the computational equipment and algorithms monitoring, their variables are represented by their sampled values which are discrete. Therefore in many cases discrete models are developed for those continuous systems, describing the relationship among their sampled variables.

A.2 The Qualitative Model Representation

Like an *ordinary* differential equation, a *qualitative* differential equation (QDE) model consists of a set of variables related by constraints. A variable represents a continuously differentiable function over the *extended* real number line, $v : \Re^* \rightarrow \Re^*$, including $\pm\infty$. However, in a *QDE* model, the range of each variable, including the independent variable time, is described qualitatively by a *quantity space*. A quantity space is a finite, totally ordered set of symbolic *landmark values* representing qualitatively important values in the real number line. Every quantity space includes landmarks for zero and positive and negative infinity.

A purely qualitative model specifies only the ordinal relations among landmarks, semi-quantitative extensions may provide bounds on the possible real values corresponding to a landmark. The algebraic and differential constraints in a QDE are simple and familiar equations, universally quantified over t, see Table A.1.

Table A.1: Algebraic and differential constraints in a QDE

(add x y z)	=	$x(t) + y(t) = z(t)$
(mult x y z)	=	$x(t).y(t) = z(t)$
(minus x y)	=	$y(t) = -x(t)$
(d/dt x y)	=	$\frac{d}{dt}x(t) = y(t)$
(constant x)	=	$\frac{d}{dt}x(t) = 0$

Since they are asserted as individual constraints, rather than composed as hierarchical expressions in traditional algebra, a QDE must include explicit variables for subexpressions. However, a QDE may also include constraints representing unknown functions in the set M^+ of monotonically increasing continuously differentiable functions, see Table A.2.

Table A.2: Monotonic constraints in a QDE

(M+ x y)	$\equiv$	$y(t) = f(x(t)), f \in M^+$
(M- x y)	$\equiv$	$y(t) = -f(x(t)), f \in M^+$

The M^+ and M^- constraints make it possible to express a QDE model including functions whose explicit form is not known, and which are only described in terms of monotonicity. An algebraic or functional constraint may specify *corresponding values*, which are tuples of landmark values known to satisfy the constraint. A QDE may also explicitly describe the boundaries of its domain of applicability by specifying *transition* conditions that carry the behaviour into a different model.

The *qualitative magnitude* of a variable is described either as a landmark value or as an open interval between two adjacent landmarks in the quantity space of that variable. The *qualitative value* of a variable is described as its qualitative magnitude and the sign of its derivative (its direction of change: inc, std or dec). A *qualitative state* of a model is a tuple of associations of qualitative values to each variable in the model.

Time is describe in the same way as every other variable. Since its direction of change is always *inc*, time progresses through an alternating sequence of landmark values (called time-points) and open intervals between adjacent time-points. The *time-points* are defined as those points in time when the qualitative state of the model changes.

A *qualitative behaviour* is a sequence of qualitative states, where each state is the immediate successor of the one before it. Because of the qualitative representation, it is possible for a finite sequence of qualitative states to represent the behaviour of a system from its initial state a $t = 0$ to a final state at $t = \infty$.

Appendix B

Fundamentals

B.1 JMLG model

B.1.1 Notation

Table B.1: JMLG model. Variables.

Symbol	Size	Description
y_t	$n_y \times 1$	Observation vector at time t
$y_{1:t}$	$n_y \times t$	Sequence of observation vectors $[y_1, y_2, \ldots, y_t]$
x_t	$n_x \times 1$	State vector at time t
$x_{0:t}$	$n_x \times t$	Sequence of state vectors $[x_1, x_2, \ldots, x_t]$
u_t	$n_u \times 1$	Input observation at time t
$u_{1:t}$	$n_u \times t$	Sequence of input observations $[u_1, u_2, \ldots, u_t]$
z_t	$n_z \times 1$	Switching state variable at time t
$z_{0:t}$	$n_z \times t$	Sequence of state vectors $[z_1, z_2, \ldots, z_t]$
γ_t	$n_x \times 1$	Process noise, $\mathcal{N}(0, I)$
v_t	$n_y \times 1$	Measurement noise, $\mathcal{N}(0, I)$

Table B.2: JMLG model. Parameters.

Symbol	Size	Description
A	$n_x \times n_x$	State transition matrix
B	$n_x \times n_\gamma$	Noise state matrix
C	$n_y \times n_x$	Output state matrix
D	$n_y \times n_\upsilon$	Noise output matrix
F	$n_x \times n_u$	Input matrix
G	$n_y \times n_u$	Null matrix
$\mathcal{Q}$	$n_x \times n_x$	State noise covariance matrix
$\mathcal{R}$	$n_y \times n_y$	Output noise covariance matrix
μ_0	$n_x \times 1$	Initial state mean
$\sum_0$	$n_x \times n_x$	Initial state noise covariance matrix
$p(z_0)$	$n_z \times 1$	Initial state probabilities for switch variable z_t
$p(z_t \mid z_{t-1})$	$n_z \times n_z$	State transition matrix for switch variable z_t

Table B.3: JMLG model. Dimensions.

Symbol	Description
n_y	Size of observation vector
n_x	Size of state vector
n_u	Size of input vector
n_z	Number of discrete modes
n_γ	Number of process noise
n_υ	Number of measurement noise
T	Length of a sequence of observation vectors

Table B.4: JMLG model. Miscellaneous.

Symbol	Description
A'	Matrix transpose of A
$\|A\|$	Determinant of A
iid	Independent and identically distributed

B.1.2 EM : Optimization algorithm

M Step

Given the probability model $p(y_{1:T}, x_{1:T}|\theta)$, we can average over $x_{1:T}$ to remove the randomness (because $x_{1:T}$ is not observed) using an averaging distribution $q(x_{1:T}|y_{1:T})$. We can define the *expected complete log likelihood*[1]

$$\mathbb{E}_{q(x_{1:T}|y_{1:T})}[l_c(\theta; y_{1:t}, x_{1:t})] \triangleq \sum_{x_{1:t}} q(x_{1:T}|y_{1:T}, \theta) \log p(y_{1:T}, x_{1:T}|\theta) \tag{B.1}$$

The M step maximizes the *expected complete log likelihood* as follows

$$\begin{aligned}
\mathcal{L}(q(x_{1:T}|y_{1:T}), \theta) &= \sum_{x_{1:t}} q(x_{1:T}|y_{1:T}) \log \frac{p(y_{1:T}, x_{1:T}|\theta)}{q(x_{1:T}|y_{1:T})} && \text{(B.2)} \\
&= \sum_{x_{1:t}} q(x_{1:T}|y_{1:T}) \log p(y_{1:T}, x_{1:T}|\theta) - \sum_{x_{1:t}} q(x_{1:T}|y_{1:T}) \log q(x_{1:T}|y_{1:T}) \\
&= \mathbb{E}_{q(x_{1:T}|y_{1:T})}[l_c(\theta; y_{1:t}, x_{1:t})] - \sum_{x_{1:t}} q(x_{1:T}|y_{1:T}) \log q(x_{1:T}|y_{1:T}) && \text{(B.3)}
\end{aligned}$$

Equation (B.3) shows that maximizing $\mathcal{L}\big(q(x_{1:T}|y_{1:T}), \theta\big)$ with respect to θ is equivalent to maximizing $\mathbb{E}_{q(x_{1:T}|y_{1:T})}[l_c(\theta; y_{1:t}, x_{1:t})]$ with respect to θ, because the second term is independent of θ.

E Step

If we use $p(x_{1:T}|y_{1:T}, \theta^{(t)})$ as an averaging distribution $q^{(t+1)}(x_{1:T}|y_{1:T})$ in equation (B.2), we have

$$\begin{aligned}
\mathcal{L}\big(p(x_{1:T}|y_{1:T}, \theta^{(t)}), \theta^{(t)}\big) &= \sum_{x_{1:t}} p(x_{1:T}|y_{1:T}, \theta^{(t)}) \log \frac{p(y_{1:T}, x_{1:T}|\theta^{(t)})}{p(x_{1:T}|y_{1:T}, \theta^{(t)})} && \text{(B.4)} \\
&= \sum_{x_{1:t}} p(x_{1:T}|y_{1:T}, \theta^{(t)}) \log p(y_{1:T}|\theta^{(t)}) && \text{(B.5)} \\
&= \log p(y_{1:T}|\theta^{(t)}) && \text{(B.6)} \\
&= l\big(\theta^{(t)}; y_{1:T}\big) && \text{(B.7)}
\end{aligned}$$

Equation (B.7) shows that $\mathcal{L}\big(q(x_{1:T}|y_{1:T}), \theta\big)$ is maximized, because $l\big(\theta; y_{1:T}\big)$ is an upper bound for $\mathcal{L}\big(p(x_{1:T}|y_{1:T}), \theta\big)$.

The *EM* algorithm uses the best possible distribution to calculate an expectation of the complete log likelihood. The M step maximizes this expected complete log likelihood with respect to the parameters to yield new values $\theta^{(t+1)}$. The E step uses a better guess $p(x_{1:T}|y_{1:T}, \theta^{(t+1)})$, which is then used as the averaging distribution in a subsequent iteration.

B.1.3 Least Squares Estimation

Notation

Table B.5 describes the ARX model's parameters and variables.

[1] $\sum_{x_{1:t}}$ means marginalization

Table B.5: Auto-Regressive with eXogenous variable model. Parameters and Variables.

Symbol	Description
y_t	Output variable at time t
u_t	Input variable at time t
a_i	Constant coefficients, auto-regressive terms
b_i	Constant coefficients, exogenous terms
n_a	Number of a_i coefficients
n_b	Number of b_i coefficients

Definitions

Some definitions accepted by the control engineering community [Ogata, 1995] are:

State. The state of a dynamic system is the smallest set of variables such that the knowledge of these variables at $t = t_0$, together with the knowledge of the input for $t \geq t_0$ completely determines the behaviour of the system for the time $t \geq t_0$.

State variables. If at least n variables $x_1, x_2, \cdots, x_n$ are needed to completely describe the behaviour of a dynamic system, then such n variables are a set of state variables. State variables need not be physically measurable or observable quantities; however, it is convenient to choose measurable quantities because optimal control laws require the feedback of all state variables.

State vector. A state vector $\mathbf{x}$ has n components, which represent the n state variables. This vector uniquely determines the system state $\mathbf{x}(t)$ for any time $t \geq t_0$, once the state at $t = t_0$ is given and the input $\mathbf{u}(t)$ for $t \geq t_0$ is specified.

State space. The n-dimensional space whose coordinate axes consist of the x_1 axis, x_2 axis, ..., x_n axis is called the state space. Any state can be represented by a point in the state space.

State space equations. There are three types of variables that are involved in the modelling of dynamic systems: *input variables, output variables,* and *state variables.* There are different state-space representations of the same system but the number of state variables is the same. The dynamic system must involve elements that memorize the values of the input for $t \geq t_1$. In a continuous-time control integrators serve as memory devices; the outputs of such integrators can be seen as the variables that define the internal state of the dynamic system. Consider a multiple-input multiple-output system involving n integrators, whose outputs are state variables $x_1(t), x_2(t), \cdots, x_n(t)$. Assume also that there are r inputs $u_1(t), u_2(t), \cdots, u_r(t)$ and m outputs $y_1(t), y_2(t), \cdots, y_m(t)$. The system may be described by

$$\begin{aligned} \dot{x}_1(t) &= f_1(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \\ \dot{x}_2(t) &= f_2(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \\ &\cdot \\ &\cdot \\ \dot{x}_n(t) &= f_n(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \end{aligned} \tag{B.8}$$

The outputs $y_1(t), y_2(t), \cdots, y_m(t)$ of the system may be given by

$$
\begin{aligned}
y_1(t) &= g_1(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \\
y_2(t) &= g_2(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \\
&\cdot \\
&\cdot \\
y_m(t) &= g_m(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t)
\end{aligned}
\tag{B.9}
$$

If we define

$$
\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \cdot \\ \cdot \\ x_n(t) \end{bmatrix}, \quad \mathbf{f}_{state}(\mathbf{x}, \mathbf{u}, t) = \begin{bmatrix} f_1(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \\ f_2(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \\ \cdot \\ \cdot \\ f_n(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \end{bmatrix} \tag{B.10}
$$

$$
\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \cdot \\ \cdot \\ y_m(t) \end{bmatrix}, \mathbf{f}_{output}(\mathbf{x}, \mathbf{u}, t) = \begin{bmatrix} g_1(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \\ g_2(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \\ \cdot \\ \cdot \\ g_n(x_1, x_2, \cdots, x_n; u_1, u_2, \cdots, u_r; t) \end{bmatrix}, \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \cdot \\ \cdot \\ u_r(t) \end{bmatrix} \tag{B.11}
$$

equations (B.8)-(B.9) become

$$
\dot{\mathbf{x}}(t) = \mathbf{f}_{state}(\mathbf{x}, \mathbf{u}, t) \tag{B.12}
$$

$$
\mathbf{y}(t) = \mathbf{f}_{output}(\mathbf{x}, \mathbf{u}, t) \tag{B.13}
$$

where the first equation is the state equation and the last is the output equation. If vector functions $\mathbf{f}_{state}$ and/or $\mathbf{f}_{output}$ involve t explicitly, then the system is called a time-varying system. If these equations are linearized about the operating state, then we have the following linearized state equation and output equation:

$$
\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{F}(t)\mathbf{u}(t) \tag{B.14}
$$

$$
\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{G}(t)\mathbf{u}(t) \tag{B.15}
$$

where $\mathbf{A}(t)$ is called the state matrix, $\mathbf{F}(t)$ is called the input matrix, $\mathbf{C}(t)$ is the output matrix, and $\mathbf{G}(t)$ the direct transmission matrix. If vector functions $\mathbf{f}$ and $\mathbf{g}$ do not involve time t explicitly then the system is called a time-invariant system:

$$
\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{F}\mathbf{u}(t) \tag{B.16}
$$

$$
\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{G}\mathbf{u}(t) \tag{B.17}
$$

This is the deterministic continuous-time version of the model shown in Chapter 3, section 3.2.1.

Observability

A represented system by Equations (B.16-B.17) has a discrete-time equivalent representation given by (for different equivalent representation see [Ogata, 1995]).

$$x_{t+1} = \mathbf{A}x_t + \mathbf{F}u_t \tag{B.18}$$

$$y_t = \mathbf{C}x_t + \mathbf{G}u_t \tag{B.19}$$

This system is completely observable if every initial state $x_{t=0}$ can be determined from the observation of y_t over a finite number of sampling periods. The system, therefore, is completely observable if every transition of the state eventually affects every element of the output vector.

The concept of observability is useful in solving the problem of reconstructing unmeasurable state variables. Complete observability means that given y_0, y_1, ..., y_t; it is possible to determine x_0[2]. This requires that the following $n_x n_y \times n_x$ matrix (called the *Observability matrix*) be of rank n_x, (see details in [Ogata, 1995] of this standard concept, also for alternative forms of the condition for complete observability).

$$\begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n_x-1} \end{bmatrix} \tag{B.20}$$

Matrix Representation

Consider the following auto-regressive with exogenous variable model where $n_a = 2$ and $n_b = 2$

$$y_t + a_1 y_{t-1} + a_2 y_{t-2} = b_0 u_t + b_1 u_{t-1} + a_2 u_{t-2} \tag{B.21}$$

If we have n_{data} data points $\{y_i, u_i\}_{i=1}^{n_{data}}$, we can predict the following observations

$$\begin{aligned}
\hat{y}_2 &= -a_1 y_1 - a_2 y_0 + b_0 u_2 + b_1 u_1 + a_2 u_0 \\
\hat{y}_3 &= -a_1 y_2 - a_2 y_1 + b_0 u_3 + b_1 u_2 + a_2 u_1 \\
\cdot &= \cdot \\
\hat{y}_{n_{data}} &= -a_1 y_{n_{data}-1} - a_2 y_{n_{data}-2} + b_0 u_{n_{data}} + b_1 u_{n_{data}-1} + a_2 u_{n_{data}-2}
\end{aligned}$$

The matrix representation is

$$\begin{bmatrix} \hat{y}_2 \\ \hat{y}_3 \\ \cdot \\ \cdot \\ \hat{y}_{n_{data}} \end{bmatrix} = \begin{bmatrix} -y_1 & -y_0 & u_2 & u_1 & u_0 \\ -y_2 & -y_1 & u_3 & u_2 & u_1 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ -y_{n_{data}-1} & -y_{n_{data}-2} & u_{n_{data}} & u_{n_{data}-1} & u_{n_{data}-2} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \end{bmatrix} \tag{B.22}$$

If we define $\mathcal{Y} = [\hat{y}_2\ \hat{y}_3\ \cdot\ \hat{y}_{n_{data}}]'$, $\Theta = [a_1\ a_2\ b_0\ b_1\ b_2]'$, it is easy to see that $\mathcal{Y} = \mathcal{X}\Theta$.

[2] x_0 could be a vector with n_x *hidden* states

Least Squares Estimation Properties

One can define equation errors as $\varepsilon = \mathcal{Y} - \mathcal{X}\theta'$. The least squares estimate of θ is defined as the vector $\widehat{\theta}$ that minimizes the loss function

$$V(\widehat{\theta}) = \frac{1}{2}\varepsilon'\varepsilon = \frac{1}{2}\|\varepsilon\|^2 \tag{B.23}$$

where $\|\varepsilon\|$ denotes the Euclidean vector norm. The corresponding minimum value of $V(\widehat{\theta})$ can be obtained when the gradient of $V(\theta)$ is equal to zero. The least squares estimate, $\widehat{\theta}$, can then be expressed as

$$\widehat{\theta}_{LS} = \left(\mathcal{X}'\mathcal{X}\right)^{-1}\mathcal{X}'\mathcal{Y} \tag{B.24}$$

Least squares optimality has several attractive features for the purposes of identification. Large errors are heavily penalized, it can be obtained by straightforward matrix algebra, and the optimization criterion is related to statistical variance, so the properties of the solution can be analyzed according to statistical criteria.

Assuming that the noise components are uncorrelated with the regressors and that $\mathbb{E}(v) = 0$ and $\mathbb{E}(v'v) = \sigma^2 I$ (white noise), the least squares estimate has the following statistical properties:

- $\widehat{\theta}_{LS}$ is an unbiased estimate of θ.
  It is easy to show that $\widehat{\theta} = (\mathcal{X}'\mathcal{X})^{-1}\mathcal{X}'(\mathcal{X}\theta + v) = \theta + (\mathcal{X}'\mathcal{X})^{-1}\mathcal{X}'v$. It then follows from the assumptions (white noise) that $\mathbb{E}[\widehat{\theta}] = \mathbb{E}[\theta + (\mathcal{X}'\mathcal{X})^{-1}\mathcal{X}'v] = \theta + (\mathcal{X}'\mathcal{X})^{-1}\mathcal{X}'\mathbb{E}[v] = \theta$.
- The covariance matrix of $\widehat{\theta}_{LS} = \sigma^2\left(\mathcal{X}'\mathcal{X}\right)^{-1}$
- An unbiased estimate of $\widehat{\sigma}^2 = \frac{2}{n_{data} - min\{n_a, n_b\}} V(\widehat{\theta})$

B.2 Particle Filtering

B.2.1 Fundamentals

Perfect Monte Carlo simulation

A set of weighted samples called particles, drawn from the posterior distribution, is used to map integrals to discrete sums. The posterior can be approximated by the following empirical estimate

$$\widehat{p}_N(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}), \tag{B.25}$$

where the random particles $\{\mathbf{x}_{0:t}^{(i)}\}_{i=1}^{N}$ are drawn from the posterior distribution and $\delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})$ denotes the Dirac-delta function. Therefore, any expectations of the form

$$\mathbb{E}\left(\mathbf{g}_t(\mathbf{x}_{0:t})\right) = \int \mathbf{g}_t(\mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{x}_{0:t} \tag{B.26}$$

can be approximated by the following estimate

$$\overline{\mathbb{E}\left(\mathbf{g}_t(\mathbf{x}_{0:t})\right)} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_t(\mathbf{x}_{0:t}^{(i)}) \tag{B.27}$$

where the particles $\mathbf{x}_{0:t}^{(i)}$ are assumed to be independent and identically distributed (*i.i.d*) for the approximation to hold. According to the law of large numbers

$$\overline{\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))} \longrightarrow_{N\to\infty}^{a.s.} \mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t})) \tag{B.28}$$

where $\longrightarrow_{N\to\infty}^{a.s.}$ means *almost sure convergence.*

If the posterior variance of $\mathbf{g}_t(\mathbf{x}_{0:t})$ is bounded $\left(var_{p(\cdot|\mathbf{y}_{1:t})}(\mathbf{g}_t(\mathbf{x}_{0:t})) < \infty\right)$ then the central limit theorem holds

$$\sqrt{N}\left(\overline{\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))} - \mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))\right) \Longrightarrow_{N\to\infty} \mathcal{N}\left(0, var_{p(\cdot|\mathbf{y}_{1:t})}(\mathbf{g}_t(\mathbf{x}_{0:t}))\right) \tag{B.29}$$

where $\Longrightarrow_{N\to\infty}$ denotes *convergence in distribution.*

Bayesian importance sampling

According to equation (B.28), as the number of particles N increases, expectations (equation B.25) can be mapped into sums. However, it is often impossible to sample directly from the posterior density function. Nevertheless, we can sample from a known proposal distribution $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ based on the following manipulation in equation (B.26):

$$\begin{aligned}
\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t})) &= \int \mathbf{g}_t(\mathbf{x}_{0:t}) \left[\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\right] q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t} \\
&= \int \mathbf{g}_t(\mathbf{x}_{0:t}) \left[\frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t})q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\right] q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t} \\
&= \int \mathbf{g}_t(\mathbf{x}_{0:t}) \left[\frac{w_t(\mathbf{x}_{0:t})}{p(\mathbf{y}_{1:t})}\right] q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}
\end{aligned} \tag{B.30}$$

where $w_t(\mathbf{x}_{0:t})$ are the unnormalized importance weights

$$w_t(\mathbf{x}_{0:t}) = \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \tag{B.31}$$

The normalizing density $p(\mathbf{y}_{1:t})$ in equation (B.30) can be eliminated as follows

$$\begin{aligned}
\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t})) &= \frac{1}{p(\mathbf{y}_{1:t})} \int \mathbf{g}_t(\mathbf{x}_{0:t}) w_t(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t} \\
&= \frac{\int \mathbf{g}_t(\mathbf{x}_{0:t}) w_t(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}}{\int p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}) \left[\frac{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\right] d\mathbf{x}_{0:t}} \\
&= \frac{\int \mathbf{g}_t(\mathbf{x}_{0:t}) w_t(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}}{\int w_t(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) d\mathbf{x}_{0:t}} \\
&= \frac{\mathbb{E}_{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}(w_t(\mathbf{x}_{0:t})\mathbf{g}_t(\mathbf{x}_{0:t}))}{\mathbb{E}_{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}(w_t(\mathbf{x}_{0:t}))}
\end{aligned} \tag{B.32}$$

We can approximate the expectations that we are interested in by drawing particles from the proposal distribution function $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$ and using the following estimate

$$\overline{\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))} = \frac{\frac{1}{N}\sum_{i=1}^{N} \mathbf{g}_t(\mathbf{x}_{0:t}^{(i)}) w_t(\mathbf{x}_{0:t}^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} w_t(\mathbf{x}_{0:t}^{(i)})} = \sum_{i=1}^{N} \mathbf{g}_t(\mathbf{x}_{0:t}^{(i)}) \widetilde{w}_t(\mathbf{x}_{0:t}^{(i)}) \tag{B.33}$$

where $\widetilde{w}_t(\mathbf{x}_{0:t}^{(i)})$ are the normalized importance weights

$$\widetilde{w}_t(\mathbf{x}_{0:t}^{(i)}) = \frac{w_t(\mathbf{x}_{0:t}^{(i)})}{\sum_{j=1}^{N} w_t(\mathbf{x}_{0:t}^{(j)})} \tag{B.34}$$

The estimate equation (B.33) is biased (ratio of estimates), but under these assumptions:

- $\mathbf{x}_{0:t}^{(i)}$ are *i.i.d* particles drawn from the proposal distribution $q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, the support of the proposal distribution includes the support of the posterior distribution, and $\mathbb{E}(\mathbf{g}_t(\mathbf{x}_{0:t}))$ exists and is finite.
- Expectations of $w_t(\mathbf{x}_{0:t}^{(i)})$ and $w_t(\mathbf{x}_{0:t}^{(i)})\mathbf{g}_t^2(\mathbf{x}_{0:t}^{(i)})$ over the posterior distribution exist and are finite. (The variance of $\mathbf{g}_t(\mathbf{x}_{0:t}^{(i)})$ and $w_t(\mathbf{x}_{0:t}^{(i)})$ must be bounded [Crisan and Doucet, 2000]).

[Doucet, 1998] demonstrated that it is possible to obtain asymptotic convergence and a central limit theorem. So the posterior density function can be approximated arbitrarily well by the point-mass estimate

$$\widehat{p}_N(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \sum_{i=1}^{N} \widetilde{w}_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}) \tag{B.35}$$

Sequential importance sampling

The proposal distribution must have the following form:

$$q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = q(\mathbf{x}_{0:t-1}|\mathbf{y}_{1:t-1})q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) \tag{B.36}$$

because we do not want to modify the previously simulated states $\mathbf{x}_{0:t-1}$ (it would be too computationally expensive). We can compute the sequential estimate of the posterior distribution over time using equation (B.36).

Considering that the model follows the Markov property

$$p(\mathbf{x}_{0:t}) = p(\mathbf{x}_0) \prod_{i=1}^{t} p(\mathbf{x}_i|\mathbf{x}_{i-1}) \tag{B.37}$$

and the observations are *i.i.d.*

$$p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t}) = \prod_{i=1}^{t} p(\mathbf{y}_i|\mathbf{x}_i) \tag{B.38}$$

The unnormalized importance weights equation can be obtained by substituting equations (B.36-B.38) into equation (B.31).

$$w_t = w_{t-1} \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})} \tag{B.39}$$

We can compute the importance weights sequentially using equation (B.39). We only have to choose the right proposal distribution $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})$ and sample from it. This procedure is known as Sequential Importance Sampling (*SIS*).

Choice of proposal distribution. This is one of the most critical design tasks in *SIS* algorithms. [Doucet, 1997] recommends proposal functions that minimize the variance of the importance weights. [Doucet *et al.*, 1999] showed that $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})$ minimizes the variance of the importance weights, so this $p(\cdot|\cdot)$ is called the optimal proposal distribution.

However, the transition prior, $q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$, is the most popular choice [Isard and Blake, 1996; Kitagawa, 1996]. The transition prior is usually easier to implement, but it has higher variance than the optimal proposal $p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})$ because it does not include the latest available information [Doucet, 1998; Liu and Chen, 1998].

Degeneracy of the SIS algorithm. We can re-write equation (B.31) and show that the variance of the importance weights increases stochastically over time.

$$\begin{aligned} w_t &= \frac{p(\mathbf{y}_{1:t}|\mathbf{x}_{0:t})p(\mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} = \frac{p(\mathbf{y}_{1:t}, \mathbf{x}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} = \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})p(\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \\ &= p(\mathbf{y}_{1:t}) \left[\frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\right] \\ &\propto \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \end{aligned} \tag{B.40}$$

The ratio[3] shown in equation (B.40) is called th *importance ratio*; [Doucet and Gordon, 1999; Kong *et al.*, 1994] showed that its variance increases over time. The degeneration or depletion of samples caused by the variance increase can be monitored through the importance weights. After a few iterations, one of the normalized importance weights tends to one and the others to zero. This problem is the main reason for the *Selection* or *Resampling* step.

B.2.2 Selection Step

Sampling Importance Resampling (*SIR*)

Re-sampling involves mapping the Dirac random measure $\{\mathbf{x}_{0:t}^{(i)}, \widetilde{w}_t^{(i)}\}$[4] into an equally weighted random measure $\{\mathbf{x}_{0:t}^{(j)}, \frac{1}{N}\}$. This can be accomplished by sampling uniformly from the discrete set $\{\mathbf{x}_{0:t}^{(i)}\}_{i=1}^N$ with probabilities $\{\widetilde{w}_t^{(i)}\}_{i=1}^N$ [Gordon *et al.*, 1993]. Figure B.1 shows a way of sampling from this discrete set. After constructing the cumulative distribution of the discrete set, a uniformly drawn sampling index i is projected onto the distribution range and then onto the distribution domain. The new sample index j is the intersection with the domain. In other words, the vector $\mathbf{x}_{0:t}^{(j)}$ is accepted as the new sample. As we can see in Figure B.1, the vectors with larger sampling weights will end up with more copies after the re-sampling process.

Sampling N times from the cumulative discrete distribution $\sum_{i=1}^N \widetilde{w}_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$ is equivalent to drawing $\{N_i\}_{i=1}^N$ from a *multinomial* distribution with parameters N and $\widetilde{w}_t^{(i)}$. [Doucet, 1998; Pitt and Shephard, 1999; Ripley, 1987] showed that this procedure can be implemented in $\mathcal{O}(N)$ operations. $var(N_i) = N\widetilde{w}_t^{(i)}(1 - \widetilde{w}_t^{(i)})$ is the variance of this procedure (*multinomial* distribution).

Residual Resampling

We can implement this scheme in three steps [Higuchi, 1997; Liu and Chen, 1998]:

1. Define $\widetilde{N}_i = \left\lfloor N\widetilde{w}_t^{(i)} \right\rfloor$
2. Perform a *Sampling Importance Resampling* (*SIR*) procedure to select the remaining $\overline{N}_t = N - \sum_{i=1}^N \widetilde{N}_i$ samples with new weights $w_t'^{(i)} = \overline{N}_t^{-1}\left(\widetilde{w}_t^{(i)} N - \widetilde{N}_i\right)$
3. Add the results to the current $\widetilde{N}_i$

[3] $p(\mathbf{y}_{1:t})$ is a constant.

[4] For simplicity, we only work with $\{\mathbf{x}_{0:t}^{(i)}, \widetilde{w}_t^{(i)}\}$ instead of $\{\mathbf{x}_{0:t}^{(i)}, z_{0:t}^{(i)}, \widetilde{w}_t^{(i)}\}$.

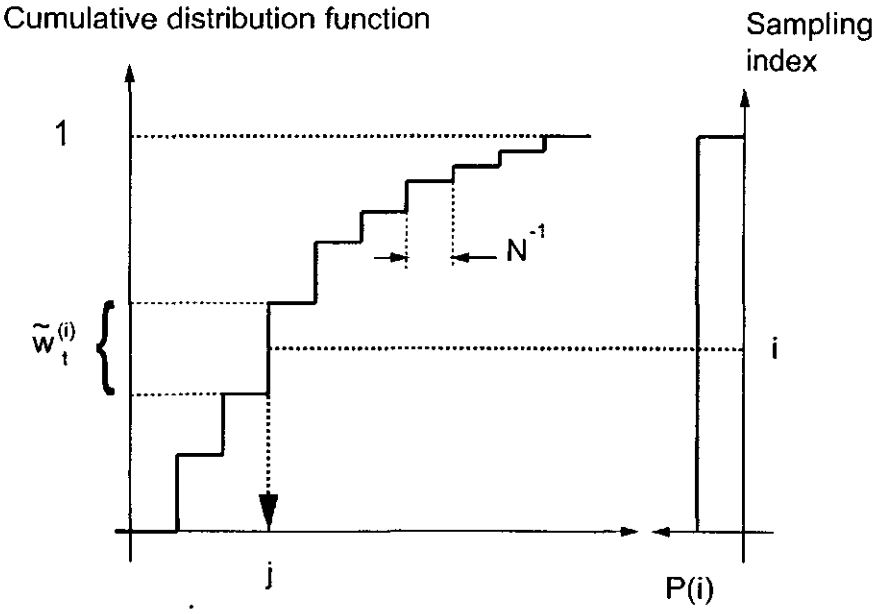

Figure B.1: Sampling Importance Resampling process

$var(N_i) = \overline{N}_t w_t'^{(i)}(1 - w_t'^{(i)})$ is the variance of this scheme, which is smaller than the one given by the *SIR* scheme.

B.3 Rao-Blackwellized Particle Filtering

B.3.1 Rao-Blackwell formula

The Rao-Blackwell theorem [Casella and Robert, 1996] shows how to improve upon any given estimator under every convex loss function. The theorem says

$$var\Big(\mathbf{f}(\mathbf{x},\mathbf{z})\Big) = var\Big(\mathbb{E}\big(\mathbf{f}(\mathbf{x},\mathbf{z})|\mathbf{z}\big)\Big) + \mathbb{E}\Big(var\big(\mathbf{f}(\mathbf{x},\mathbf{z})|\mathbf{z}\big)\Big) \tag{B.41}$$

where $\mathbf{f}(\mathbf{x},\mathbf{z})$ is an estimator of $\mathbf{x}$ and $\mathbf{z}$. Hence, $var\big(\mathbb{E}(\mathbf{f}(\mathbf{x},\mathbf{z})|\mathbf{z})\big) \leq var\big(\mathbf{f}(\mathbf{x},\mathbf{z})\big)$, so we can conclude that $\mathbb{E}(\mathbf{f}(\mathbf{x},\mathbf{z})|\mathbf{z})$ is a lower variance estimator. If we can generate particles of $\mathbf{z}$ and analytically evaluate the expectation of $\mathbf{x}$ given $\mathbf{z}$, we will need less particles for a given accuracy.

If we sample in a small discrete space ($\mathbf{z}$), instead of a large hybrid space ($\mathbf{z}$, $\mathbf{x}$), the performance has to be much better.

B.3.2 Variance reduction

If we were able to sample N *i.i.d* random particles, $\{\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}^{(i)}\}_{i=1}^{N}$, according to $p(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, then an empirical estimate of this distribution would be given by

$$\widehat{p}_N(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta_{\left(\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}^{(i)}\right)}(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}) \tag{B.42}$$

where $\delta_{\left(\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}^{(i)}\right)}(\mathbf{z}_{0:t}, \mathbf{x}_{0:t})$ denotes the Dirac delta function located at $(\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}^{(i)})$. The expected value of any function $\mathbf{g}_t$ of the hidden variables with respect to this distribution, $\mathbb{E}(\mathbf{g}_t)$, using

$$\mathbb{E}(\mathbf{g}_t) = \int \mathbf{g}_t(\mathbf{z}_{0:t}, \mathbf{x}_{0:t})\widehat{p}_N(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t})d\mathbf{z}_{0:t}d\mathbf{x}_{0:t} \tag{B.43}$$

could be approximated by the following estimate

$$\overline{\mathbb{E}(\mathbf{g}_t)} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_t(\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}^{(i)}) \tag{B.44}$$

this estimate is unbiased, and follows conditions given by equations (B.28-B.29). Using the *Bayesian importance sampling* method (Section B.2.1) we can introduce an arbitrary easy-to-sample importance distribution $q(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. Then

$$\mathbb{E}(\mathbf{g}_t) = \frac{\mathbb{E}_{q(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\left(\mathbf{g}_t(\mathbf{z}_{0:t}, \mathbf{x}_{0:t})w_t(\mathbf{z}_{0:t}, \mathbf{x}_{0:t})\right)}{\mathbb{E}_{q(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\left(w_t(\mathbf{z}_{0:t}, \mathbf{x}_{0:t})\right)} \tag{B.45}$$

where the importance weight is equal to

$$w_t(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = \frac{p(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \tag{B.46}$$

Given N *i.i.d* samples $\left\{\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}^{(i)}\right\}_{i=1}^{N}$ distributed according to $q(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t})$, a Monte Carlo estimate of $I(\mathbf{g}_t)$ is given by the following (see equations B.33-B.34):

$$\overline{\mathbb{E}^1(\mathbf{g}_t)} = \frac{\overline{A^1(\mathbf{g}_t)}}{\overline{B^1(\mathbf{g}_t)}} = \frac{\frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_t(\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}^{(i)})w_t(\mathbf{z}_{0:t}^{(i)}\mathbf{x}_{0:t}^{(i)})}{\frac{1}{N}\sum_{i=1}^{N}w_t(\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}^{(i)})} = \sum_{i=1}^{N}\widetilde{w}_t^{(i)}\mathbf{g}_t(\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}^{(i)}) \tag{B.47}$$

where the normalized importance weights $\widetilde{w}_t^{(i)}$ are equal to

$$\widetilde{w}_t^{(i)} = \frac{w_t(\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}^{(i)})}{\sum_{j=1}^{N}w_t(\mathbf{z}_{0:t}^{(j)}, \mathbf{x}_{0:t}^{(j)})} \tag{B.48}$$

$\overline{\mathbb{E}^1(\mathbf{g}_t)}$ is biased for finite N, but converges toward $\mathbb{E}(\mathbf{g}_t)$, equations (B.28-B.29). Consider the case where we can marginalize out $\mathbf{x}_{0:t}$ analytically. We can propose an alternative estimate for $\mathbb{E}(\mathbf{g}_t)$ with reduced variance. As $p(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t}) = p(\mathbf{z}_{0:t}|\mathbf{y}_{1:t})p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}, \mathbf{z}_{0:t})$ (Rao-Blackwell formula, Section B.3.1), where $p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}, \mathbf{z}_{0:t})$ is a distribution that can be computed exactly, then an approximation of $p(\mathbf{z}_{0:t}|\mathbf{y}_{1:t})$ yields straightforwardly an approximation of $p(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}|\mathbf{y}_{1:t})$. If $\mathbb{E}_{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}, \mathbf{z}_{0:t})}\left(\mathbf{g}_t\left(\mathbf{z}_{0:t}, \mathbf{x}_{0:t}\right)\right)$ can be evaluated in a closed-form expression, then the following alternative importance sampling estimate of $\mathbb{E}(\mathbf{g}_t)$ can be used

$$\overline{\mathbb{E}^2(\mathbf{g}_t)} = \frac{\overline{A^2(\mathbf{g}_t)}}{\overline{B^2(\mathbf{g}_t)}} = \frac{\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{p\left(\mathbf{x}_{0:t}|\mathbf{y}_{1:t}, \mathbf{z}_{0:t}^{(i)}\right)}\mathbf{g}_t\left(\mathbf{z}_{0:t}^{(i)}, \mathbf{x}_{0:t}\right)w_t\left(\mathbf{z}_{0:t}^{(i)}\right)}{\frac{1}{N}\sum_{i=1}^{N}w_t\left(\mathbf{z}_{0:t}^{(i)}\right)} \tag{B.49}$$

where

$$w_t(\mathbf{z}_{0:t}) = \frac{p(\mathbf{z}_{0:t}|\mathbf{y}_{1:t})}{q(\mathbf{z}_{0:t}|\mathbf{y}_{1:t})} \tag{B.50}$$

The following two propositions demonstrate [Doucet *et al.*, 2000a] that $\overline{\mathbb{E}^2(\mathbf{g}_t)}$ will require a reduced number N of samples over $\overline{\mathbb{E}^1(\mathbf{g}_t)}$, as we only need to sample from a lower-dimensional distribution.

Proposition 1 *The variances of the importance weights, the numerators and the denominators satisfy for any N*

$$var_{q(\mathbf{z}_{0:t}|\mathbf{y}_{1:t})}\big(w(\mathbf{z}_{0:t})\big) \leq var_{q(\mathbf{z}_{0:t},\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\big(w(\mathbf{z}_{0:t},\mathbf{x}_{0:t})\big) \tag{B.51}$$

$$var_{q(\mathbf{z}_{0:t}|\mathbf{y}_{1:t})}\big(\overline{A^2(\mathbf{g}_t)}\big) \leq var_{q(\mathbf{z}_{0:t},\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\big(\overline{A^1(\mathbf{g}_t)}\big) \tag{B.52}$$

$$var_{q(\mathbf{z}_{0:t}|\mathbf{y}_{1:t})}\big(\overline{B^2(\mathbf{g}_t)}\big) \leq var_{q(\mathbf{z}_{0:t},\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\big(\overline{B^1(\mathbf{g}_t)}\big) \tag{B.53}$$

[Bernardo and Smith, 1994] showed that if $var_{p(\mathbf{z}_{0:t},\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\big(\mathbf{g}_t(\mathbf{z}_{0:t},\mathbf{x}_{0:t})\big) < +\infty$ and $w(\mathbf{z}_{0:t},\mathbf{x}_{0:t}) < +\infty$ for any $(\mathbf{z}_{0:t},\mathbf{x}_{0:t})$, then $\overline{\mathbb{E}^1(\mathbf{g}_t)}$ satisfies the Central Limit Theorem (*CLT*). $\overline{\mathbb{E}^2(\mathbf{g}_t)}$ also satisfies a *CLT*.

Proposition 2 *Under the assumption given above,* $\overline{\mathbb{E}^1(\mathbf{g}_t)}$ *and* $\overline{\mathbb{E}^2(\mathbf{g}_t)}$ *satisfy a* CLT

$$\sqrt{N}\left(\overline{\mathbb{E}^1(\mathbf{g}_t)} - \mathbb{E}(\mathbf{g}_t)\right) \Longrightarrow_{N\to\infty} \mathcal{N}(0,\sigma_1^2) \tag{B.54}$$

$$\sqrt{N}\left(\overline{\mathbb{E}^2(\mathbf{g}_t)} - \mathbb{E}(\mathbf{g}_t)\right) \Longrightarrow_{N\to\infty} \mathcal{N}(0,\sigma_2^2) \tag{B.55}$$

where $\sigma_1^2 \geq \sigma_2^2$, σ_1^2 *and* σ_2^2 *being given by*

$$\sigma_1^2 = \mathbb{E}_{q(\mathbf{z}_{0:t},\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}\left[\left((\mathbf{g}_t(\mathbf{z}_{0:t},\mathbf{x}_{0:t}) - \mathbb{E}(\mathbf{g}_t))w_t(\mathbf{z}_{0:t},\mathbf{x}_{0:t})\right)^2\right] \tag{B.56}$$

$$\sigma_2^2 = \mathbb{E}_{q(\mathbf{z}_{0:t}|\mathbf{y}_{1:t})}\left[\left((\mathbb{E}_{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t},\mathbf{z}_{0:t})}(\mathbf{g}_t(\mathbf{z}_{0:t},\mathbf{x}_{0:t})) - \mathbb{E}(\mathbf{g}_t)w_t(\mathbf{z}_{0:t})\right)^2\right] \tag{B.57}$$

Appendix C

Look-ahead Rao-Blackwellized Particle Filtering

C.1 System Identification

C.1.1 General Procedure of System Identification

An *identification experiment* is performed by exciting the system and observing its input and output over a time interval. These signals are recorded in a computer. It is important to consider the application context, i.e. the intended *purpose of modelling*, such as, simulation models, fault detection models, and control system analysis.

The *design of experiments* includes selection and determination of several factors, like input signals, sampling time, identification time, open or closed loop identification, off-line or on-line identification, equipment for the signal generation, signal filtering, etc. The most theoretical part of the system identification procedure is the *selection of model*. A priori knowledge and engineering intuition and insight must be combined with mathematical properties of the models.

The unknown parameters of the model are then determined using some statistically based *parameter estimation* method. To measure the goodness of the model fit a criterion needs to be specified. Often the sum of squares of error signals (residuals) is used as a criterion. In practice, the selection of structure and the estimation of parameters are often done iteratively. Finally, *model validation* means the testing whether the estimated model is sufficiently good for the intended use of the model.

C.1.2 Preliminary experiments

A standard procedure for identification is to start with some preliminary experiments; further experiments are then conducted for model estimation. The preliminary experiments are aimed at the rough modelling of the process and can provide a basic qualitative understanding of the system. In general two types of simple experiments are considered:

- *Step response techniques* [Smith and Corripio, 1997] to test the range of linearity of the process, and to find the static gains and a rough estimation of the largest relevant time constant. Step change signals are applied to the selected candidate process inputs. The time interval of the step change should allow the process to reach

its steady state. The steady state linearity of the process then can be checked by fitting a basic model. Here one can find nonlinear behaviour.

- *White noise experiments* can be applied for determining the process bandwidth and time delays. The inputs are supplied with mutually independent white noise signals. The estimations are based on the spectral and correlation analysis of input and output signals [Ljung, 1987; Zhu and Backx, 1993]. An appropriate broad band periodic signal, either a PRBS (pseudo random binary sequence) or multi-sine, can also be applied.

C.1.3 Design of input signals

The input signals used in an identification experiment can have a significant effect on the resulting parameter estimates. The design of input signals involves both the selection of the form of the signal and the choice of its amplitude.

In practice most processes are nonlinear and linear models can hence only be an approximation. A linearization of the nonlinear dynamics will be valid only in some region. To estimate the parameters of the linearized model, too large an input amplitude should be avoided.

On the other hand, an improvement in the accuracy of the estimates can be expected when the input amplitude is increased. This is natural since by increasing the input, the signal-to-noise ratio will increase and the disturbances will have less impact. A rule of thumb for choosing a minimum amplitude is that the effect of the input on the output in a diagram must at least be perceptible to the eye.

Good parameter identification requires the application of a frequency-rich input signal. In order to guarantee that the estimation algorithm has unique solutions, some minimum requirement should be imposed on the test signals. This is called the persistent excitation condition. A single variable discrete signal is said to be persistent exciting of order n if the following limit exists:

$$R_{uu}(\tau) = \lim_{N\to\infty} \sum_{t=1}^{N} u(t)u(t-\tau) \tag{C.1}$$

where the matrix

$$\begin{bmatrix} R_{uu}(0) & R_{uu}(1) & \dots & R_{uu}(n-1) \\ R_{uu}(-1) & R_{uu}(0) & & \vdots \\ \vdots & & \ddots & \\ R_{uu}(1-n) & \dots & & R_{uu}(0) \end{bmatrix}$$

is positive definite and consequently, invertible. It can be shown that a necessary condition for consistent estimation of an n^{th} order linear process is that the test signal is persistently exciting of order $2n$. The step function based on Equation (C.1) is persistently exciting of order 1 only. *Sum of sinusoid signals, filtered white noise signals* and *PRBS* are the most frequently used in identification practice. We used *PRBS* signals; we describe them in the following section.

Pseudo Random Binary Sequence (PRBS)

Pseudo random binary sequences are sequences of rectangular pulses, modulated in width, that approximate a discrete-time white noise and thus have a spectral content rich in frequencies. They have produced excellent results in the control engineering community[1]. They are called pseudo random because they are characterized by a

[1]Personal communication with Ian D. Landau, 1989.

sequence length within which the variations in pulse width vary randomly; however, they are periodic over a large time horizon, the period being defined by the length of the sequence.

The PRBS can be generated by means of shift registers with feedback. The maximum length of a sequence is $2^N - 1$ in which N is the number of cells of the shift register. Figure C.1 shows the generation of a PRBS of length 31 ($= 2^5 - 1$) obtained using a 5-cell shift register, Figure C.2. Table C.1 [Landau, 1990] shows the structure enabling a maximum length PRBS to be generated for different numbers of cells. A simple choice for the clock period of the PRBS generator T_{PRBS} is to use the sampling time $T_{sampling}$ of the system. The maximum duration of a PRBS pulse is equal to the number of cells [Landau, 1990] (a very important characteristic).

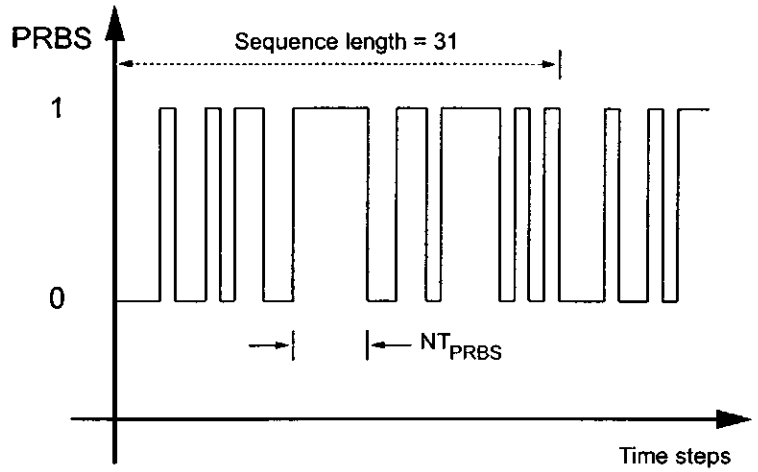

Figure C.1: PRBS of length 31 generated by means of a 5-cells shift register. After the 31st time-step the PRBS becomes periodic. The maximum duration of a PRBS pulse is equal to $5T_{PRBS}$.

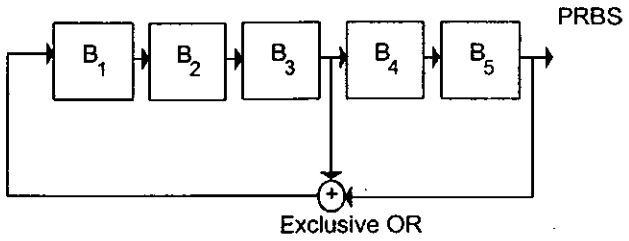

Figure C.2: PRBS generator with 5-cells shift register. Using an exclusive OR, bits 3 and 5 must be added and fed-back to bit 1, then bit 1 moves to bit 2, and so on.

In order to correctly identify the steady-state gain of the process, the duration of at least one of the PRBS pulses must be greater than the rise or settling time $T_{settling}$ of the process, see Figure C.3.

This condition can result in fairly large values of N corresponding to a sequence length of prohibitive duration. Therefore, in a large number of practical situations, a multiple of the sampling time $T_{sampling}$ is chosen as the clock period for the PRBS generator (it is recommended that a multiplier less than or equal to 4 be chosen).

Choosing the sampling interval involves a trade-off. These are the primary considerations:

- In many cases the final application of the model defines the possible choices of sampling interval.

Table C.1: Generation of maximum length PRBS. This table gives the structure enabling a maximum length PRBS to be generated for different numbers of cells, i.e. using a 5-cell shift register (adding 3 and 5 bits), the maximum length PRBS is 31.

Number of cells	Sequence length	Bits to add	
2	3	1	2
3	7	1	3
4	15	3	4
5	31	3	5
6	63	5	6
7	127	4	7
8	255	2	8
9	511	5	9
10	1023	7	10
11	2047	9	11

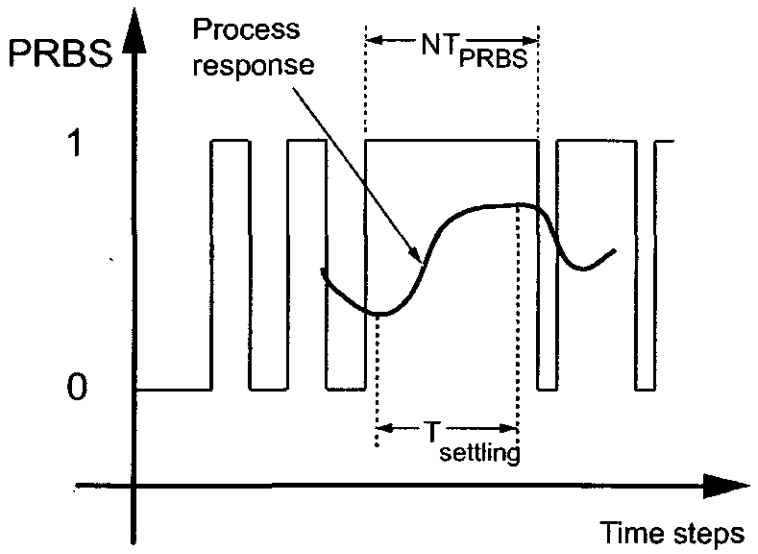

Figure C.3: Condition for correct identification of the steady-state process gain. The duration of at least one of the PRBS pulses must be greater than the rise or settling time $T_{settling}$ of the process.

- The information loss due to sampling is best described in the frequency domain. It is well known that a sinusoid with a frequency higher than half of the sampling frequency cannot, when sampled, be distinguished from a sinusoid under this frequency. Consequently the part of the signal spectrum that corresponds to higher frequencies will be interpreted as contributions from lower frequencies. This is the so called alias phenomenon. It also means that the spectrum of the sampled signal will be a superposition of different parts of the original spectrum.

- As a rule of thumb, 5 to 15 samples per settling time $T_{settling}$ should be taken. It may often be much worse to select the sampling interval too large than too small.

Another aspect to consider when using a PRBS test signal is that in order to cover the entire frequency spectrum generated by a particular PRBS, the length of the test must be at least equal to the length of the sequence. In general, using periodic signals such as a PRBS, the duration of the test should be a number of complete periods or sequences [Landau, 1990].

Once data have been collected, there are several possible deficiencies that can appear, such as

- high frequency disturbances in the data record
- bursts and outliers
- drifts
- offset and low frequency disturbances
- numerical values of different signals are not in the same order of magnitude
- presence of significant time delays
- etc

see [Morales-Menéndez, 1992] for more details.

C.1.4 Some guidelines for model validation

Model validation means checking whether the learned model agrees with the real process behaviour. Experimental validation is the most important test, i.e. to evaluate whether the inference problem that motivated the learning model task can be solved using the obtained learned model. However it is often impossible, costly or dangerous to test all possible learned models with their intended use, especially when we are trying to infer faults. Instead, confidence in the learned model is developed in other ways. The two main methods are:

1. *Verification of a priori assumptions.*
   - Linearity: comparison of learned models obtained for different input amplitudes. Comparison of learned models with measured transient functions in both directions.
   - Time variance: comparison of models built from different training data sets.
   - Residuals must be statistically independent with zero expectation, and independent from the input signal.
   - Input signals must be persistently excited, without outliers.
   - Variance and covariance matrices of the parameter estimates must decrease with increasing number of samples.
2. *Verification of the input-output behaviour.* A final judgement of the learned model is obtained by comparing the real data and predicted input-output behaviour. This can be done by:
   - Comparison of the real data, $\{y_i\}_{i=1}^{n_{data}}$, and the synthetic data, $\{\widehat{y}_i\}_{i=1}^{n_{data}}$ generated by the *Jump Markov Linear Gaussian* model. Using the training data and different types of input signals.

- Comparison of the cross correlation function based on the real data and on synthetic data generated by the *JMLG* model.
- Cross-checking, i.e. the verification of the learned model using different testing data.

C.2 Rao-Blackwell theorem. Fundamentals

We presented a short definition of the *Rao-Blackwell* theorem in Appendix B; however, we are including a complete description here.

Based on the *Rao-Blackwell* theorem, we can say that sufficient statistics play an important role in finding good estimators for parameters. If $\hat{\theta}$ is an unbiased estimator for θ and if U is a statistic that is sufficient for θ then there is a function of U that is also an unbiased estimator for θ and has no larger variance than $\hat{\theta}$. If we seek unbiased estimators with small variances, we can restrict our search to estimators that are functions of sufficient statistics (as in *la-RBPF*).

Rao-Blackwell Theorem. Let $\hat{\theta}$ be an unbiased estimator for θ such that $V(\hat{\theta}) < \infty$. If U is a sufficient statistic for θ, define $\hat{\theta}^* = \mathbb{E}(\hat{\theta}|U)$. Then for all θ,

$$\mathbb{E}(\hat{\theta}^*) = \theta \quad and \quad V(\hat{\theta}^*) \leq V(\hat{\theta})$$

Proof. Since U is sufficient for θ, the conditional distribution of any statistic, given U, does not depend on θ. Thus $\hat{\theta}^* = \mathbb{E}(\hat{\theta}|U)$ is not a function of θ and is therefore a statistic. Since $\hat{\theta}$ is an unbiased estimator for θ, it implies that

$$\mathbb{E}(\hat{\theta}^*) = \mathbb{E}\big(\mathbb{E}(\hat{\theta}|U)\big) = \mathbb{E}(\hat{\theta}) = \theta$$

Thus, $\hat{\theta}^*$ is an unbiased estimator for θ. Also

$$\begin{aligned} V(\hat{\theta}) &= V\big(\mathbb{E}(\hat{\theta}|U)\big) + \mathbb{E}\big(V(\hat{\theta}|U)\big) \\ &= V\big(\hat{\theta}^*\big) + \mathbb{E}\big(V(\hat{\theta}|U)\big) \end{aligned}$$

Since $V(\hat{\theta}|U = u) \geq 0$ for all u, it follows that $\mathbb{E}\big(V(\hat{\theta}|U)\big) \geq 0$ and therefore $V(\hat{\theta}^*) \leq V(\hat{\theta})$. □

This theorem implies that an unbiased estimator for θ with a small variance is a function of a sufficient statistic. Since many statistics are sufficient for a parameter θ associated with a distribution, we have to define which sufficient statistic should be applied. For the distributions that we use in this research, the *factorization criterion* typically identifies a statistic U that best summarizes the information in the data about the parameter θ. Such statistics are called *minimal sufficient statistics*.

These statistics have an important property (*completeness*) guaranteeing that, if we apply the Rao-Blackwell theorem using U, we obtain an unbiased estimator for θ with *Minimum Variance*. Such an estimator is called a *Minimum Variance Unbiased Estimator* (*MVUE*) [Casella and Berger, 1990].

Direct computation of conditional expectations can be difficult. However, if U is the sufficient statistic that best summarizes the data and some function of U, $f(U)$, can be found such that $\mathbb{E}\big(f(U)\big) = \theta$, it follows that $f(U)$ is the *MVUE* for θ.

Factorization criterion. Let U be a statistic based on the random sample $Y_1, Y_2, \ldots, Y_n$. Then U is a *sufficient statistic* for the estimation of parameter θ if and only if the likelihood L can be factored into two nonnegative functions,

$$L(y_1, y_2, \ldots, y_n|\theta) = g(u, \theta) \times h(y_1, y_2, \ldots, y_n) \tag{C.2}$$

where $g(u, \theta)$ is a function only of u and θ and $h(y_1, y_2, \ldots, y_n)$ is not a function of θ.

Appendix D

Experimental Implementation

D.1 Industrial dryer

D.1.1 Diagnosis/estimation tests

Tables (D.1-D.3) show other representative designs of random sequences for the industrial dryer. We generated 1,000 data points for each, where air temperature was the observable variable $\{y_t\}$ for each faulty condition $\{z_t\}$. Figures (D.1-D.2) compare the experimental and the simulated results (JMLG model) for these random sequences. Random sequences No. 2 and No. 4 contain simultaneous faults.

Table D.1: Industrial dryer. Random sequence No 2.

Step	Interval	z_t	Model name	Fan speed	Dryer grill	Exit vent
1	(1 , 204)	1	Normal operation	low	opened	clear
2	(205, 266)	2	Faulty fan	high	opened	clear
3	(267, 310)	3	Faulty grill	low	closed	clear
4	(311, 334)	4	Faulty fan and grill	high	closed	clear
5	(335, 894)	3	Faulty grill	low	closed	clear
4	(895, 1,000)	2	Faulty fan	high	opened	clear

D.1.2 Results

Figures (D.3-D.5) show diagnosis error results for random sequences No. 2, 3, and 4.

Variation in the transition matrix

Table D.4 shows the diagnosis error mean for different percentages of variation in transition matrix probabilities $p(z_t|z_{t-1})$ for the standard *RBPF* algorithm. Table D.5 shows the corresponding information for *la-RBPF*.

Table D.2: Industrial dryer. Random sequence No. 3.

Step	Interval	z_t	Model name	Fan speed	Dryer grill	Exit vent
1	(1 , 421)	1	Normal operation	low	opened	clear
2	(422, 686)	3	Faulty fan	low	closed	clear
3	(687, 721)	1	Normal operation	low	opened	clear
4	(722, 846)	2	Faulty fan	high	opened	clear
5	(847, 974)	1	Normal operation	low	opened	clear
6	(975, 1,000)	3	Faulty grill	low	closed	clear

Table D.3: Industrial dryer. Random sequence No. 4.

Step	Interval	z_t	Model Name	Fan speed	Dryer grill	Exit vent
1	(1 , 222)	1	Normal operation	low	opened	clear
2	(223, 274)	2	Faulty fan	high	opened	clear
3	(275, 475)	1	Normal operation	low	opened	clear
4	(476, 534)	2	Faulty fan	high	opened	clear
5	(535, 810)	1	Normal operation	low	opened	clear
6	(811, 876)	4	Faulty fan and grill	high	closed	clear
7	(877, 916)	3	Faulty grill	low	closed	clear
8	(917, 936)	1	Normal operation	low	opened	clear
9	(937, 1,000)	4	Faulty fan and grill	low	closed	clear

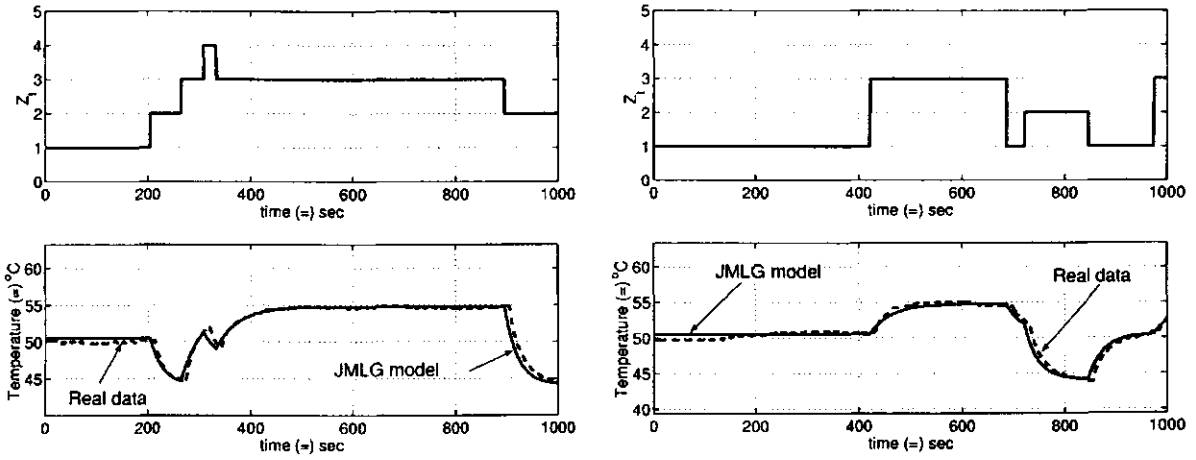

Figure D.1: Industrial dryer. Random sequence No. 2 and No. 3. The upper graphs show the discrete mode over time, while the lower graphs show the corresponding real temperature and synthetic data (*JMLG* model). The left graphs are random sequence No. 2; the right graphs are random sequence No. 3.

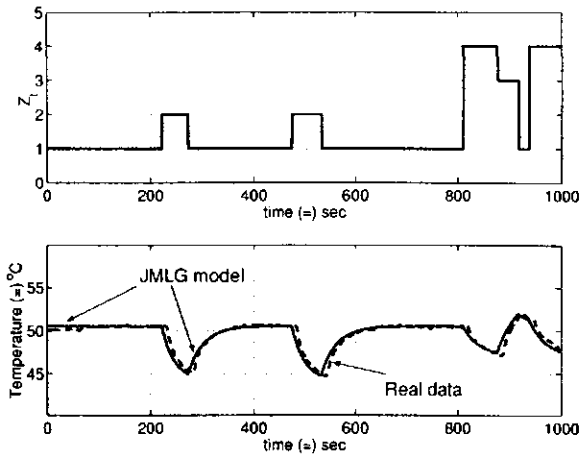

Figure D.2: Industrial dryer. Random sequence No. 4. The upper graph shows the discrete mode over time; the lower graph shows the real temperature and the synthetic data.

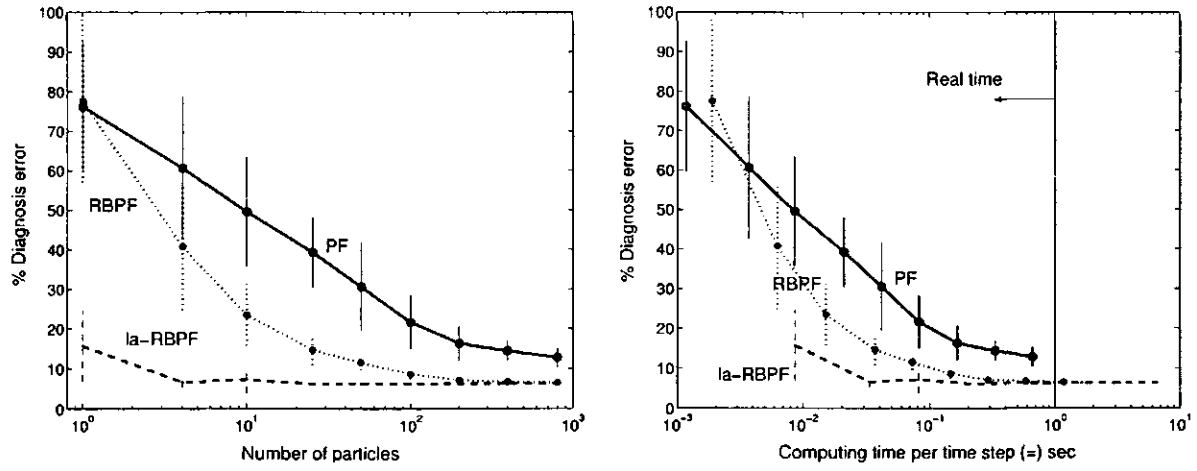

Figure D.3: Industrial dryer. Diagnosis error. Random sequence No. 2. The left graph shows diagnosis error versus number of particles, while the right graph shows diagnosis error versus computing time per time step.

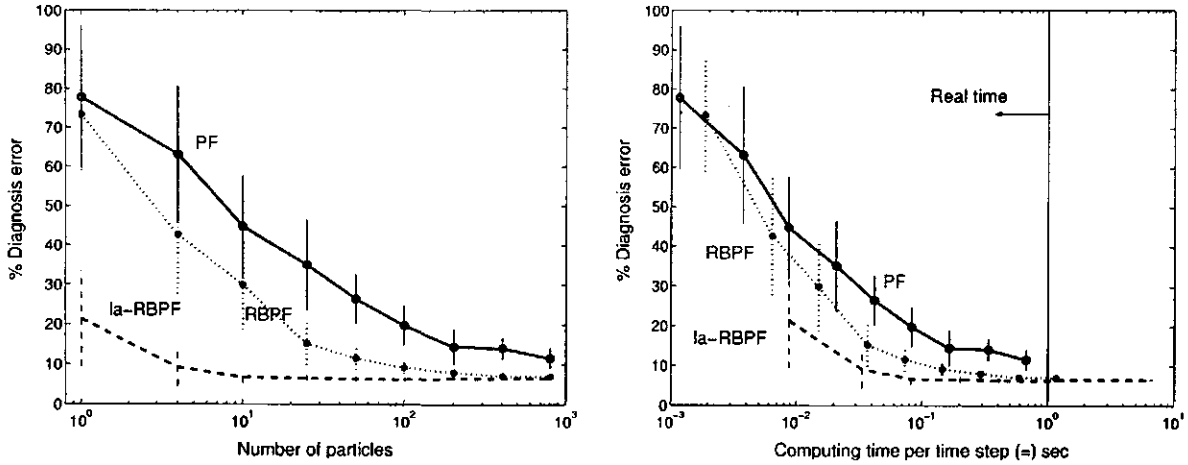

Figure D.4: Industrial dryer. Diagnosis error. Random sequence No. 3.

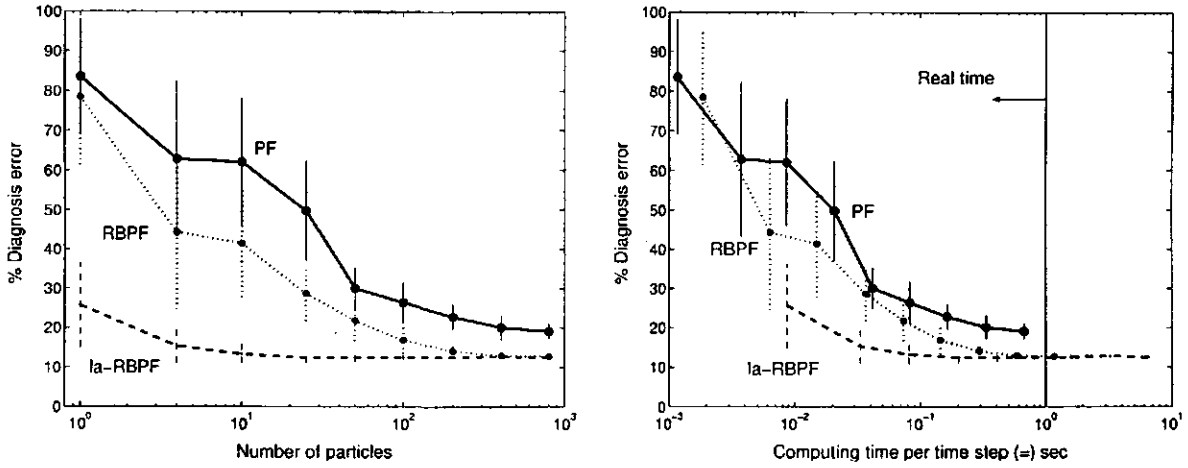

Figure D.5: Industrial dryer. Diagnosis error. Random sequence No. 4.

Table D.4: Diagnosis error for variations in $p(z_t|z_{t-1})$ using RBPF

Particles	Percentage of variation							
N	0%	5%	10%	20%	40%	80%	160%	320%
1	78.3	78.3	72.6	72.0	74.0	71.9	72.5	72.8
4	66.2	64.6	61.1	63.6	55.7	41.4	38.5	24.2
10	31.7	29.9	29.5	29.1	27.0	27.7	20.2	17.1
25	21.9	21.5	17.7	16.8	15.4	14.5	11.3	11.0
50	19.8	17.3	16.7	13.7	12.4	11.6	9.5	9.7
100	8.5	8.4	8.2	8.2	9.1	8.4	7.9	8.2
200	8.2	8.1	8.1	8.0	8.0	7.7	7.9	8.2

Table D.5: Diagnosis error for variations in $p(z_t|z_{t-1})$ using la-RBPF

Particles	Percentage of variation							
N	0%	5%	10%	20%	40%	80%	160%	320%
1	27.38	27.34	27.08	27.72	27.78	28.10	28.58	25.42
4	19.12	18.98	19.00	18.64	18.04	17.68	14.92	9.02
10	12.04	12.04	12.04	12.06	10.32	10.36	9.30	8.58
25	7.80	7.70	7.64	8.78	7.74	7.62	7.82	7.78
50	6.80	6.78	6.76	6.94	7.00	7.20	7.18	7.16
100	6.86	7.02	7.00	7.06	6.82	7.00	7.04	7.26
200	6.70	6.78	6.76	6.80	6.84	7.00	7.02	7.20

D.2 Level tank

D.2.1 Diagnosis/estimation tests

The left graph in Figure D.6 shows a random sequence for the level-tank (4 discrete modes); the right graph shows another random sequence for 5 discrete modes.

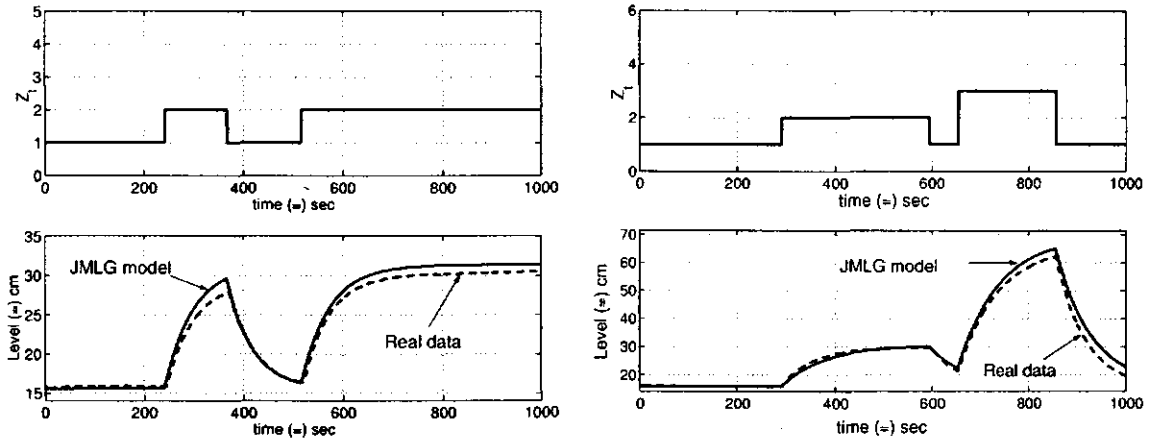

Figure D.6: Level tank. Random sequences and *JMLG* model performance. The upper graphs show the discrete mode over time. The lower graphs show the corresponding real tank level and synthetic data (*JMLG* model). The left graphs are for 4 discrete modes; the right graphs are for 5 discrete modes.

D.2.2 Results

Table D.6 shows the numerical results for the three Particle Filtering algorithms using different numbers of particles for random sequence No. 1, (Chapter 5, Figure 5.12).

Figure D.7 shows diagnosis error versus number of particles, and diagnosis error versus computing time per time-step, respectively, for random sequence No. 1 (Chapter 3, section 5.3.3). This data was collected using more

Table D.6: Level tank (4 discrete modes). Random sequence No. 1.

Particles	Diagnosis error, %, mean			Diagnosis error, %, SD		
N	PF	RBPF	la-RBPF	PF	RBPF	la-RBPF
1	76.10	64.72	37.51	17.69	21.08	15.09
4	76.62	57.89	15.16	16.08	23.47	15.52
10	69.78	57.15	17.49	13.23	23.71	17.72
25	64.02	49.39	7.89	14.69	22.89	8.62
50	54.81	42.16	4.42	10.34	23.82	2.35
100	49.87	34.13	3.19	11.91	23.44	2.08
200	47.54	31.35	2.14	8.94	23.73	0.95
400	40.49	23.60	1.84	13.08	22.68	0.23
800	30.91	13.32	1.81	14.75	15.60	0.10
1600	20.92	8.44	1.77	16.52	10.57	0.06

precise sensors[1]. Note the difference between this figure and Figure 5.10.

Figure D.8 shows diagnosis error versus number of particles, and diagnosis error versus computing time per time step, respectively, for random sequence No. 2, in which noisy sensor-transmitters were used.

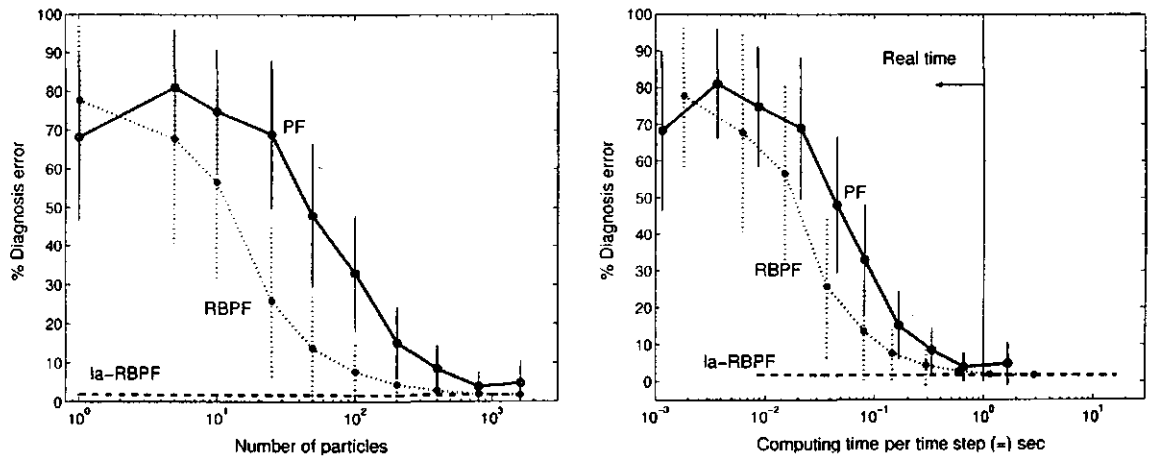

Figure D.7: Level tank (4 discrete modes). Diagnosis error. Random sequence No. 1.

Figures (D.9-D.10) show diagnosis error versus number of particles, and diagnosis error versus computing time per time step, respectively, for random sequences No. 2 and No. 3 using 5 discrete modes. For these tests, well calibrated sensor-transmitters were used.

[1]Sensor-transmitters were re-calibrated.

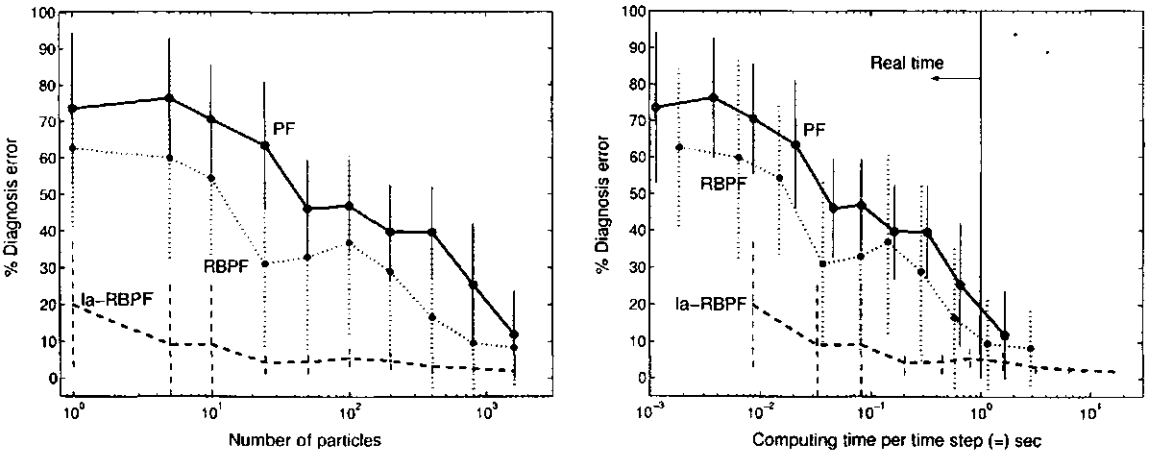

Figure D.8: Level tank (4 discrete modes). Diagnosis error. Random sequence No. 2.

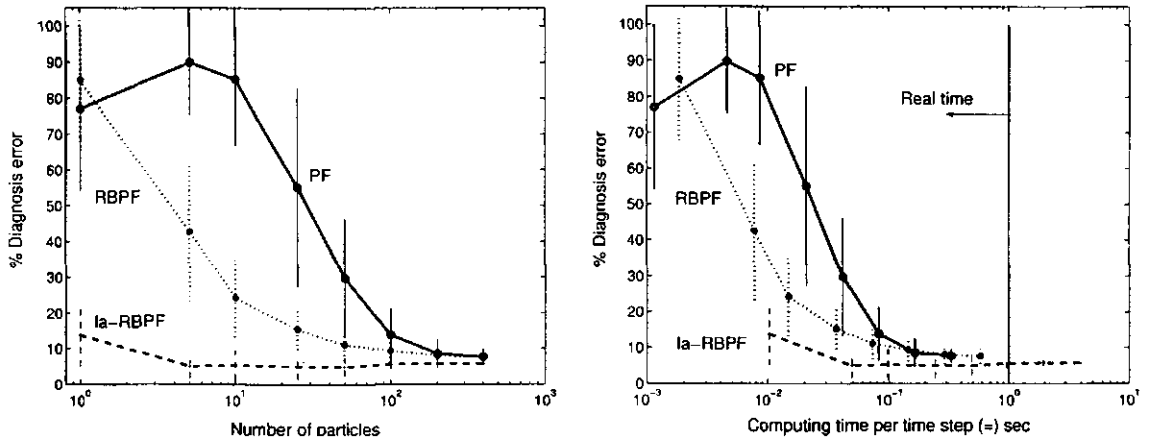

Figure D.9: Level tank (5 discrete modes). Diagnosis error. Random sequence No. 2.

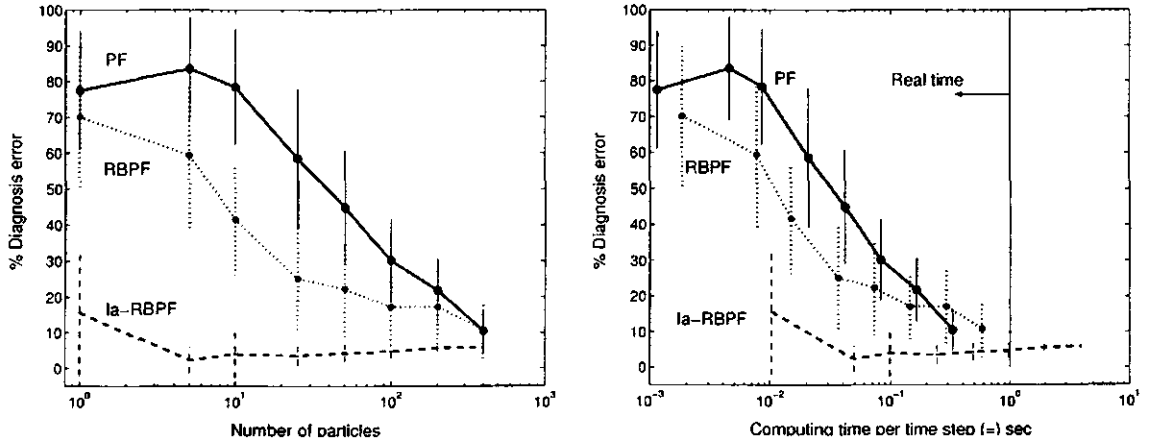

Figure D.10: Level tank (5 discrete modes). Diagnosis error. Random sequence No. 3.

D.3 Industrial heat exchanger

D.3.1 Modelling

We present some intermediate results of the proposed procedure for learning the model parameters of one discrete mode. Except for the *Expectation-Maximization* (*EM*) method, we are using standard identification tools from the control engineering community.

First, we show a step change test and its results using the *EM* alogrithm. Then, we implement a *Pseudo Random Binary Sequence* (*PRBS*) test and learn the parameters using the *Least Squares Estimation* (*LSE*) method followed by the *EM* method.

Step response test

The following Figure D.11 shows a step response in one discrete mode for this domain. The upper graph indicates that the chosen input variable was increased 15 % and then decrease 15 %, while the lower graph indicates the output variable response. The process is slightly nonlinear. As we can see in the lower graph, the responses to the positive and negative steps are different. They have different transient responses and different final steady states. However, for simplicity in this discrete mode, we nevertheless considered it a linear system. This decision could increase diagnosis error.

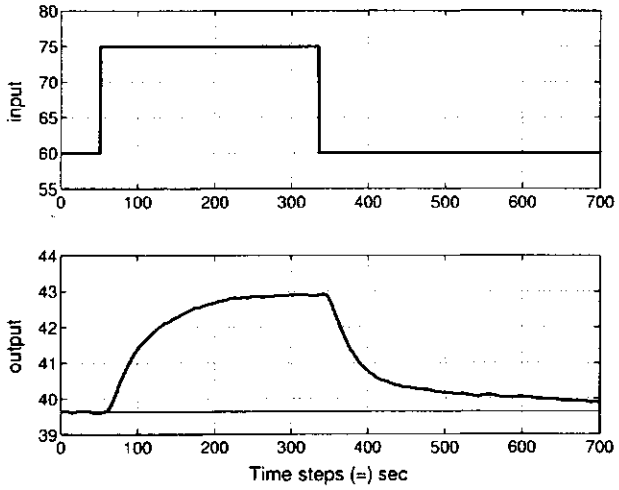

Figure D.11: Heat exchanger. Step response. The upper graph shows the input step change in both directions. The lower graph shows the transient response. As we can see, the dynamic responses are different. Also, the final steady state is different from the initial one, with different settling time.

Expectation-Maximization Method using a Step Change test

We use the *Expectation-Maximization* method to model the stochastic state space representation of each discrete mode. Figures (D.12-D.13) show some intermediate results. In this example, we tried two values for the continuous state space dimension, n_x. The two left plots in Figure D.12 show the results when $n_x = 1$ (one dimension). These plots represent log likelihood and convergence versus number of iterations, respectively; at most 100 iterations were

permitted. The right plots in Figure D.12 show the input and output variables used in the *EM* algorithm. Note that we used deviation variables in both cases. The real output and the simulated data (*JMLG* model) are compared in the lower right plot. The modelling procedure gives good results using only one dimension ($n_x = 1$).

Figure D.13 shows the results when $n_x = 2$. By simple visual analysis, we can see that one dimension for the continuous state space is enough; it gives better results than $n_x = 2$.

Note that we must try different values of n_x for each discrete mode, using a different experimental data set, in order to get each of the n_z individual models that form the *JMLG* model.

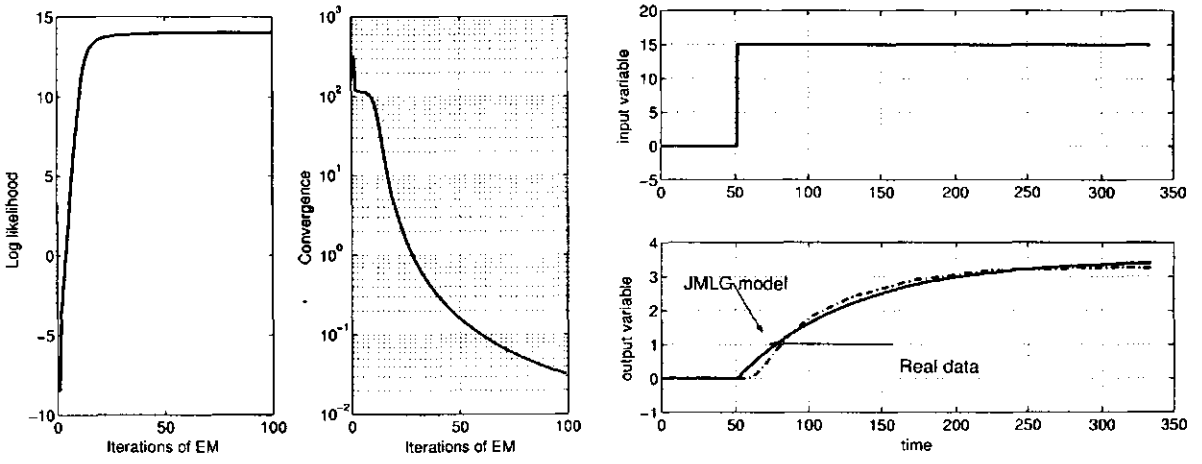

Figure D.12: Expectation-Maximization Method, $n_x = 1$. The two left graphs represent log likelihood and convergence versus number of iterations, respectively. We model the data using $n_x = 1$ dimension for the continuous state variables. The right graphs show the real input and output data versus time. The lower right graph also compares the real and simulated data (*JMLG* model).

PRBS test / Least Squares Estimation

The following Figure D.14 shows an experimental PRBS test. The left graph shows the data generated by the PRBS. This PRBS test was implemented using a 9-cell shift register. 545 data points were collected every 10 sec. The upper left plot shows the input signal, while the lower left plot shows the output signal. Both signals are deviation variables. The right graph compares the real data and the synthetic data. The synthetic data were generated using an ARX model. The ARX parameters were learned by the *Least Squares Estimation* method. For clarity, we plot only 100 data points; the rest of the comparison has the same pattern.

Figure D.15 shows the autocorrelation function of residuals (upper graphs) and the cross correlation (lower graphs) between the residuals and the input. All these response curves should be *small*. In both cases, we show 99 % confidence regions (shadowed areas) around the zero limits for these values. For a model to pass the residual test, the curves should ideally be inside these regions. However, for practical purposes a region between $[-0.2, +0.2]$ (95 % confidence) is enough. The correlation functions are given up to lag 25.

The left graphs in Figure D.15 show a model structure which fails the test; in the right graphs the model passes the residual test. We considered a 95 % confidence region, $[-0.2, +0.2]$.

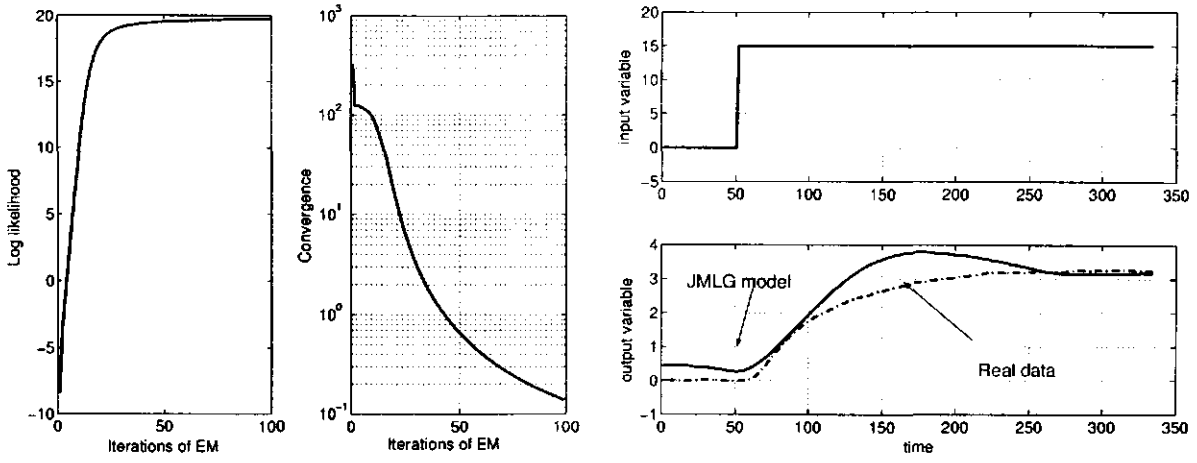

Figure D.13: Expectation-Maximization Method, $n_x = 2$. The two left graphs represent log likelihood and convergence versus number of iterations, respectively. We used $n_x = 2$ dimensions for the continuous state variables. The right graphs show the real input and output data versus time. The lower right graph also compares the real and simulated data (*JMLG* model).

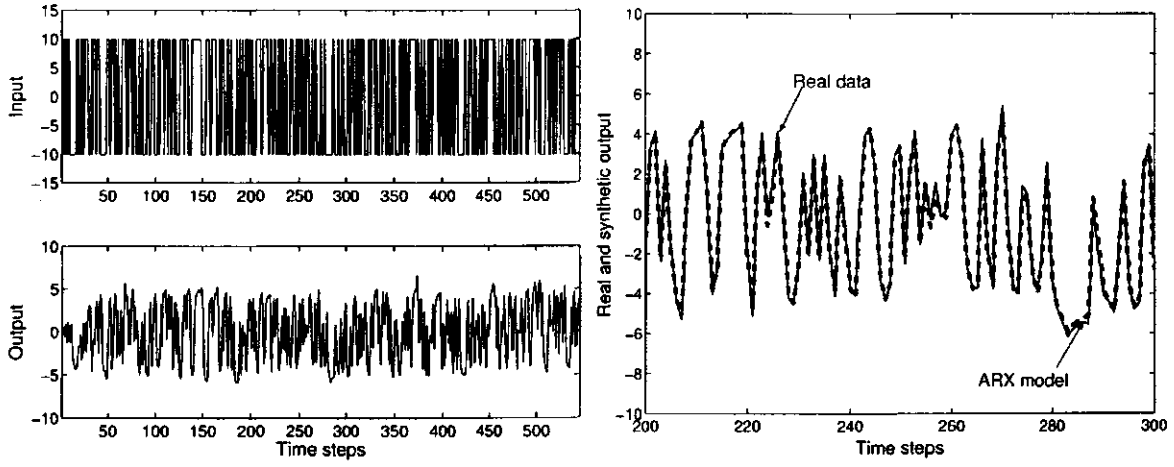

Figure D.14: Pseudo Random Binary Sequence test. A PRBS test was designed using a 9-cell shift register; 545 data points were collected every 10 sec. The left graphs show the input and output signals (deviation variables). The right graph compares the real and synthetic data (ARX model) for this data set.

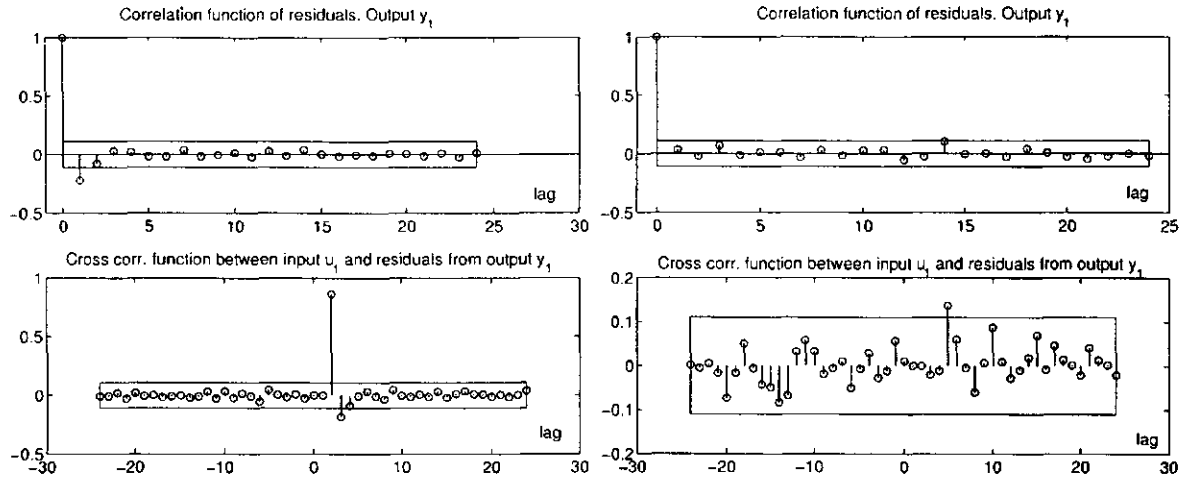

Figure D.15: Residuals test. The upper plots show the autocorrelation function of residuals, and the lower plots show the cross correlation. The shadowed areas represent 99 % confidence regions around zero limits for these values. The correlation functions are given up to lag 25. The left graphs show a model that failed the test, while in the right graphs the test was passed, using 95 % confidence.

Expectation-Maximization Method using a PRBS test

The ARX model allows us to build a deterministic model, where the signals are corrupted by noise. In order to get a stochastic model for this process, we must use the *EM* method with the same experimental data generated by the PRBS test, but also learn the noise matrices. We use the learned parameters for the deterministic state space model as an initial approximation.

Figure D.16 shows preliminary results. In this example, we used one dimension for continuous state space variables, $n_x = 1$. The two left plots represent log likelihood and convergence versus number of iterations, respectively. The number of iterations was limited to 100, and the convergence to 0.00001. The right plots show a small interval of the input and output variables. The complete input/output data set is shown in Figure D.14. The real output and the simulated data (*JMLG* model) are compared in the lower right plot.

D.3.2 Diagnosis/estimation tests

D.3.3 Results

Variation in the transition matrix

Table D.7 shows the diagnosis error mean for different percentage of variations in probabilities of the most representative discrete mode of the transition matrix $p(z_t|z_{t-1})$ for the *la-RBPF* algorithm. Variation in the less the less representative discrete mode of the transition matrix $p(z_t|z_{t-1})$ and in the prior distribution of the discrete mode do not show.

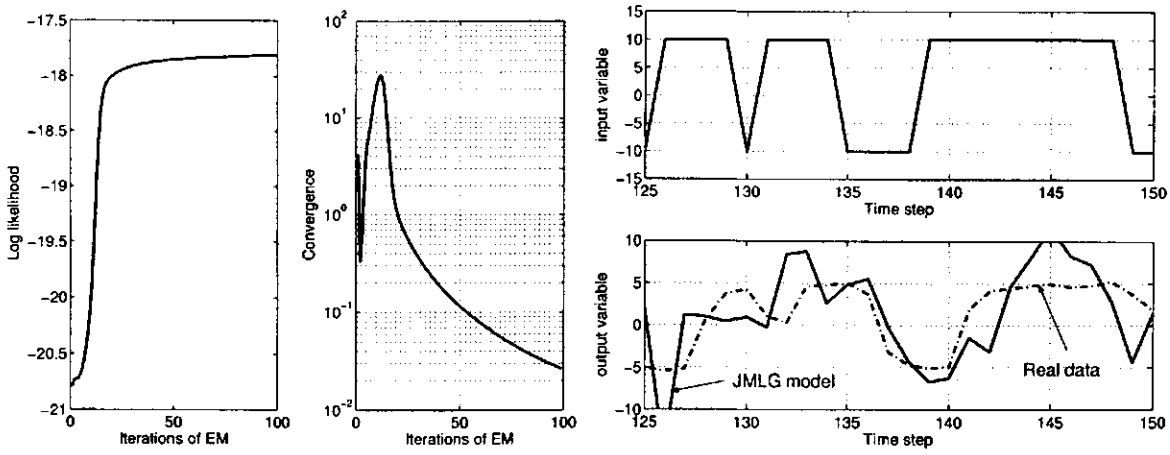

Figure D.16: Expectation-Maximization Method using a PRBS test. The two left graphs represent log likelihood and convergence versus number of iterations, respectively. We model this data using $n_x = 1$ dimension for the continuous state space variables. The right graphs show the real input and output data per time step. The lower right graph also compares the real and simulated output (*JMLG* model).

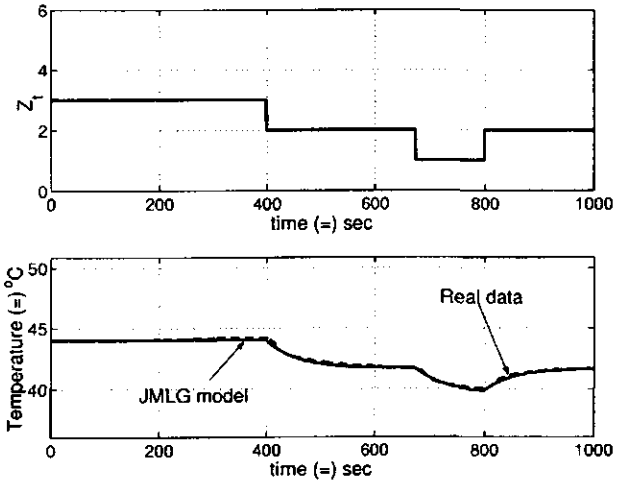

Figure D.17: Heat exchanger. Random sequence No. 2.

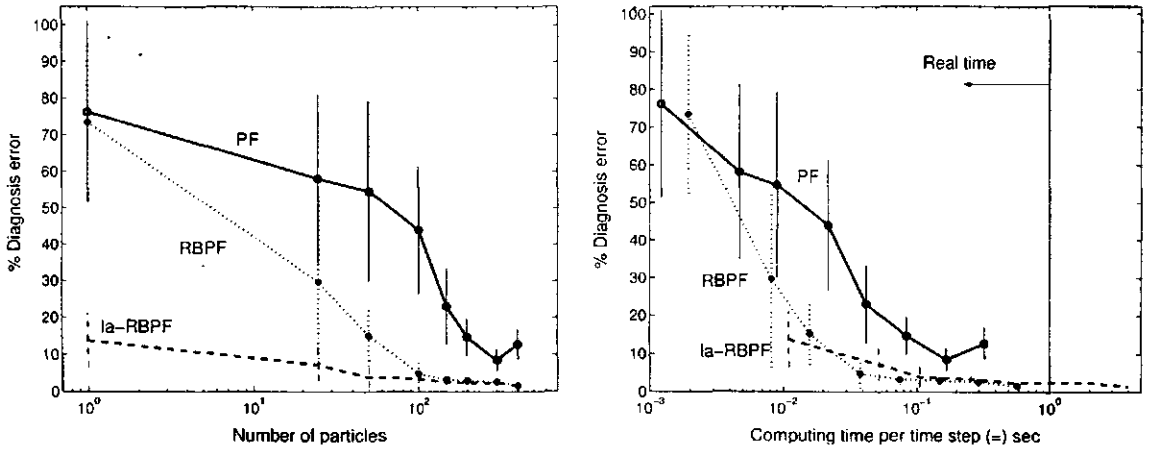

Figure D.18: Heat exchanger. Diagnosis error. Random sequence No. 2.

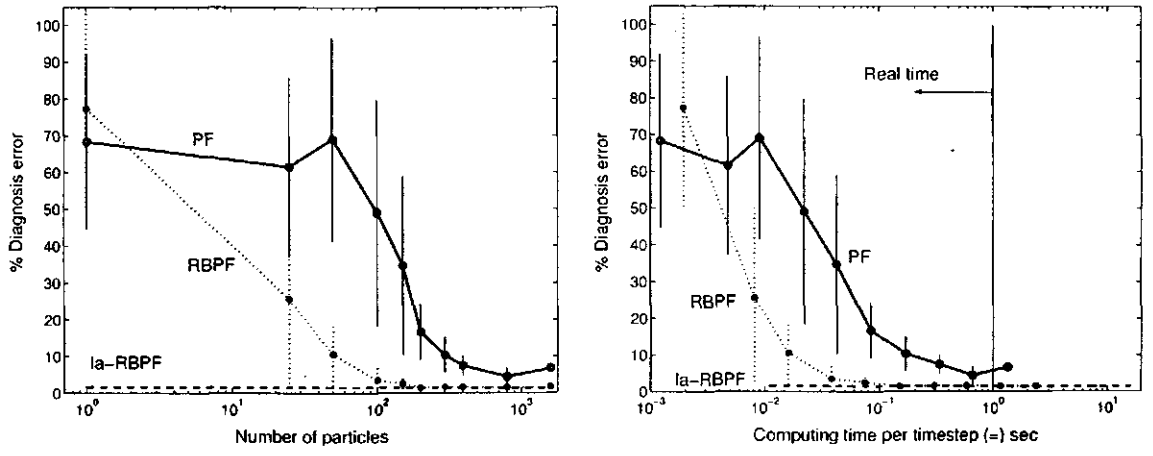

Figure D.19: Heat exchanger. Diagnosis error. Random sequence No. 3.

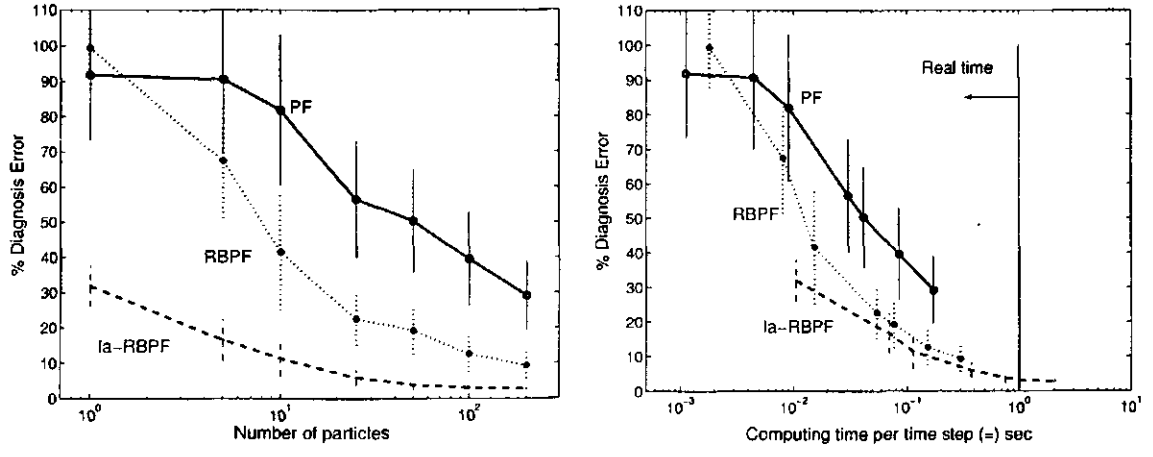

Figure D.20: Heat exchanger. Diagnosis error. Random sequence No. 4.

Table D.7: Diagnosis error for variations in $p(z_t|z_{t-1})$ using la-RBPF

Particles	Percentage of variation							
N	0%	5%	10%	20%	40%	80%	160%	320%
1	38.72	38.72	41.42	41.42	41.42	40.70	40.70	35.60
4	10.40	10.40	10.40	10.40	10.24	9.06	9.06	9.00
10	8.44	8.44	8.44	8.20	8.12	8.06	8.18	7.86
25	5.16	5.10	4.98	5.00	5.00	4.96	4.76	4.76
50	5.48	5.48	5.48	5.48	5.06	5.02	5.14	5.00
100	4.06	4.04	4.08	4.08	4.40	4.42	4.44	4.18
200	3.18	3.18	3.18	3.16	3.16	3.06	3.04	3.22

D.4 Mobile robot

D.4.1 Results

Figures (D.21-D.23) show some results from the smooth and tiled floors. Diagnosis error is plotted versus number of particles and computing time per time step.

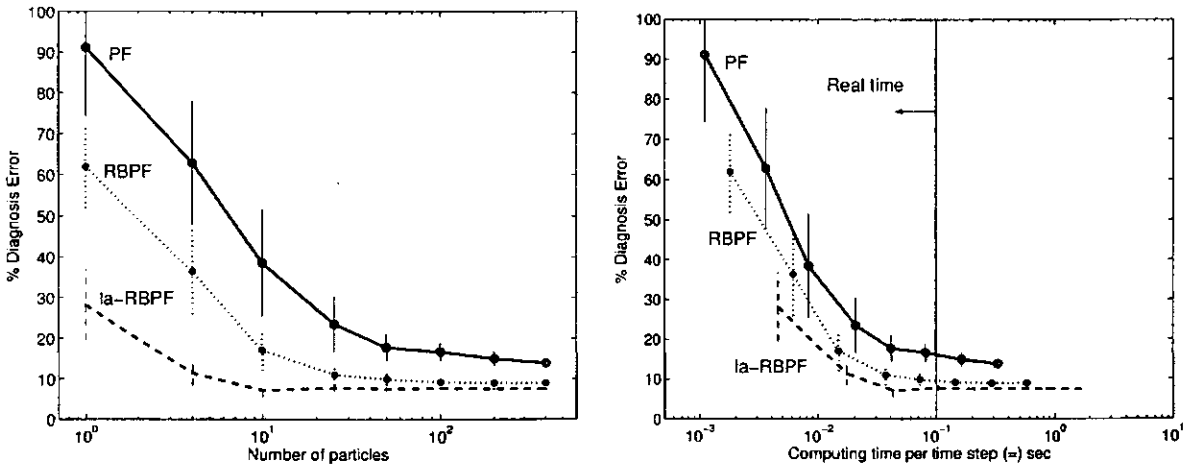

Figure D.21: Random sequence, smooth floor.

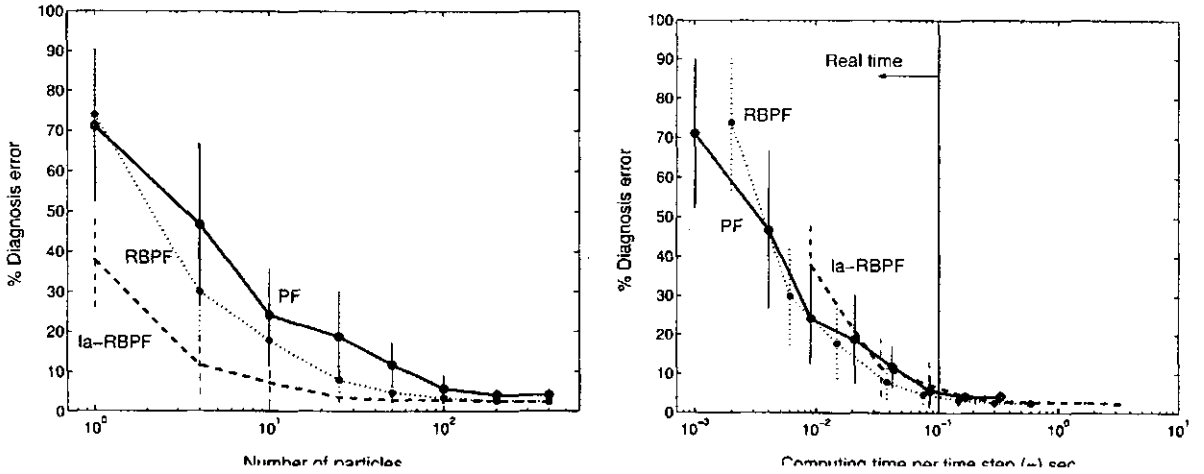

Figure D.22: Random sequence, tiled floor.

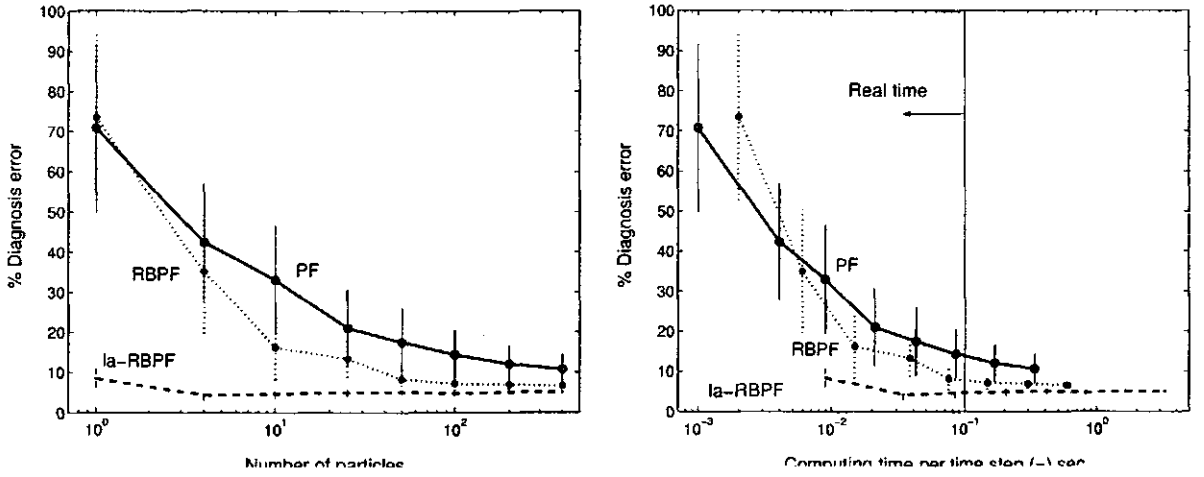

Figure D.23: Random sequence, tiled floor (filtered signal).

Appendix E

Simulated Systems

E.1 Continuous Stirred Tank Reactor (CSTR)

In the jacketed chemical reactor CSTR, described in Chapter 6 section 6.2, a second-order exothermic reaction $2A \rightarrow B$ takes place. The reaction rate constant was considered to follow the Arrhenius equation (E.1). Table E.1 defines the variables in this system, and gives their values at steady state.

$$k(T_r(t)) = k_0 e^{\left(\frac{-\alpha}{T_r(t)+460}\right)} \tag{E.1}$$

E.1.1 Nonlinear model

The following equations represent a simplified non-linear model:

Mass Balance on A

$$V\rho\frac{dC_A(t)}{dt} = W(C_{Ai}(t) - C_A(t)) - kC_A^2(t)V\rho \tag{E.2}$$

Enthalpy Balance on reacting mass

$$V\rho C_P\frac{dT_r(t)}{dt} = W(t)C_p(T_{ri}(t) - T_r(t)) - UA(T_r(t) - T_{jo}(t)) + (-\Delta H)VkC_A^2(t) \tag{E.3}$$

Enthalpy Balance on Jacket

$$M_j C_{pj}\frac{dT_{jo}(t)}{dt} = UA(T_r(t) - T_{jo}(t)) - W_j(t)C_{pj}(T_{jo}(t) - T_{ji}(T)) \tag{E.4}$$

E.1.2 Linear model

In this system, we consider that the following variables change over time: $C_A(t)$, $T_r(t)$, $C_{Ai}(t)$, $W(t)$, $T_{ji}(t)$, $T_{ri}(t)$, $T_{jo}(t)$ and $W_j(t)$. The mass balance on A given by equation (E.2) can be linearized, as we show:

$$\frac{dC_A(t)}{dt} = \frac{W(t)}{V\rho}(C_{Ai}(t) - C_A(t)) - k_0 e^{[-\alpha/(T_r(t)+460)]}C_A^2(t) \tag{E.5}$$

$$\frac{dC_A(t)}{dt} = \frac{W(t)C_{Ai}(t)}{V\rho} - \frac{W(t)C_A(t)}{V\rho} - k_0 e^{[-\alpha/(T_r(t)+460)]}C_A^2(t)$$

$$\frac{dC_A(t)}{dt} = f_1[W(t), C_{Ai}(t)] + f_2[W(t), C_A(t)] + f_3[T_r(t), C_A(t)] \tag{E.6}$$

Table E.1: Variables.

Var	Value	Units	Definitions
C_A	3.5955	lb/ft^3	Concentration of reactant A in reactor and exit stream
C_{Ai}	10.8	lb/ft^3	Concentration of reactant A in feed
a	2,560	^oR	Constant in Arrhenius expression for reaction rate
k	0.0278	$ft^3/lb-min$	Reaction rate constant
k_o	1.43	$ft^3/lb-min$	Constant in Arrhenius expression
$-\Delta H$	867	Btu/lb_A	Heat of reaction
C_p	0.9	Btu/lb^oF	Specific heat of reacting mixture
C_{pj}	1.0	Btu/lb^oF	Specific heat of water
A	500	ft^2	Effective jacket transfer area
ρ	80	lb/ft^3	Density of reacting mixture
U	1.2	$Btu/min\ ft^2\ ^oF$	Heat transfer coefficient
T_{ri}	150	^oF	Input reactants temperature
Tr	190.0611	^oF	Reactor temperature
T_{rm}	190.0611	^oF	Measured reactor temperature
T_{jo}	120.0222	^oF	Outlet Jacket temperature
T_{ji}	80	^oF	Inlet Jacket temperature
V	250	ft^3	Reactor volume
W	1,000	lb/min	Feed mass flow rate
W_j	1,050	lb/min	Water cooling rate at jacket
M_j	4,000	lb	Mass of jacket water

The linear model of each term (f_1, f_2 and f_3) in the previous equation is obtained by Taylor series expansion. We only present the detailed derivation for term f_1:

$$f_1[W(t), C_{Ai}(t)] = f_1[\overline{W}, \overline{C}_{Ai}] + \frac{\partial f_1[\overline{W}, \overline{C}_{Ai}]}{\partial W(t)}(W(t) - \overline{W}) + \frac{\partial f_1[\overline{W}, \overline{C}_{Ai}]}{\partial C_{Ai}(t)}(C_{Ai}(t) - \overline{C}_{Ai}) \quad \text{(E.7)}$$

$$f_1[W(t), C_{Ai}(t)] = \frac{\overline{W}\overline{C}_{Ai}}{V\rho} + \frac{\overline{C}_{Ai}(W(t) - \overline{W})}{V\rho} + \frac{\overline{W}(C_{Ai}(t) - \overline{C}_{Ai})}{V\rho} \quad \text{(E.8)}$$

$$f_1[W(t), C_{Ai}(t)] = -\frac{\overline{W}\overline{C}_{Ai}}{V\rho} + \frac{\overline{C}_{Ai}}{V\rho}W(t) + \frac{\overline{W}}{V\rho}C_{Ai}(t) \quad \text{(E.9)}$$

where $\overline{W} = W(t=0)$ and $\overline{C}_{Ai} = C_{Ai}(t=0)$. The linear model for f_2 is:

$$f_2[W(t), C_A(t)] = -\frac{\overline{W}\overline{C}_A}{V\rho} + \frac{\overline{C}_A}{V\rho}W(t) + \frac{\overline{W}}{V\rho}C_A(t) \quad \text{(E.10)}$$

where $\overline{C}_A = C_A(t=0)$. Finally, the linear model for f_3 is:

$$
\begin{aligned}
f_3[T_r(t), C_A(t)] &= f_3[\overline{T}_r, \overline{C}_A] + \frac{\partial f_3[\overline{T}_r, \overline{C}_A]}{\partial T_r(t)}(T_r(t) - \overline{T}_r) + \frac{\partial f_3(\overline{T}_r, \overline{C}_A)}{\partial C_A(t)}(C_A(t) - \overline{C}_A) \qquad \text{(E.11)} \\
k(\overline{T}_r) &= k_0 e^{-a/(\overline{T}_r + 460)} \\
f_3[T_r(t), C_A(t)] &= k(\overline{T}_r)\overline{C}_A^2 + \frac{a\overline{C}_A^2 k(\overline{T}_r)}{(\overline{T}_r + 460)^2}(T_r(t) - \overline{T}_r) + 2\overline{C}_A k(\overline{T}_r)(C_A(t) - \overline{C}_A) \\
f_3[T_r(t), C_A(t)] &= -k(\overline{T}_r)\overline{C}_A^2\left[1 + \frac{a\overline{T}_r}{(\overline{T}_r + 460)^2}\right] + \frac{ak(\overline{T}_r)\overline{C}_A^2}{(\overline{T}_r + 460)^2}T_r(t) + 2k(\overline{T}_r)\overline{C}_A C_A(t) \qquad \text{(E.12)}
\end{aligned}
$$

where $\overline{T}_r = T_r(t = 0)$. The original nonlinear differential equation (E.2) (mass balance on A) becomes

$$
\begin{aligned}
\frac{dC_A(t)}{dt} &= \alpha_1 + \alpha_{11}C_A(t) + \alpha_{12}T_r(t) + \beta_{11}C_{Ai}(t) + \beta_{13}W(t) \qquad \text{(E.13)} \\
\alpha_1 &= \frac{\overline{W}(\overline{C}_A - \overline{C}_{Ai})}{V\rho} + k(\overline{T}_r)\overline{C}_A^2\left[1 + \frac{a\overline{T}_r}{(\overline{T}_r + 460)^2}\right] \\
\alpha_{11} &= -\left(\frac{\overline{W}}{V\rho} + 2k(\overline{T}_r)\overline{C}_A\right) \\
\alpha_{12} &= \frac{-ak(\overline{T}_r)\overline{C}_A^2}{(\overline{T}_r + 460)^2} \\
\beta_{11} &= \frac{\overline{W}}{V\rho} \\
\beta_{13} &= \frac{\overline{C}_{Ai} - \overline{C}_A}{V\rho}
\end{aligned}
$$

Using a similar procedure, equation (E.3), which describes the enthalpy balance in the reactor, becomes

$$
\begin{aligned}
V\rho C_P\frac{dT_r(t)}{dt} &= W(t)C_p(T_{ri}(t) - T_r(t)) - UA(T_r(t) - T_{jo}(t)) + (-\Delta H)Vk(T_r(t))C_A^2(t) \\
\frac{dT_r(t)}{dt} &= \alpha_2 + \alpha_{21}C_A(t) + \alpha_{22}T_r(t) + \alpha_{23}T_{jo}(t) + \beta_{22}T_{ri}(t) + \beta_{23}W(t) \qquad \text{(E.14)} \\
\alpha_2 &= \frac{\overline{W}(\overline{T}_r - \overline{T}_{ri})}{V\rho} - \frac{(-\Delta H)k(\overline{T}_r)\overline{C}_A^2}{\rho C_p}\left[1 + \frac{a\overline{T}_r}{(\overline{T}_r + 460)^2}\right] \\
\alpha_{21} &= \frac{2(-\Delta H)k(\overline{T}_r)\overline{C}_A}{\rho C_p} \\
\alpha_{22} &= -\left[\frac{\overline{W}}{V\rho} - \frac{a(-\Delta H)k(\overline{T}_r)\overline{C}_A^2}{\rho C_p(\overline{T}_r + 460)^2} + \frac{UA}{V\rho C_p}\right] \\
\alpha_{23} &= \frac{UA}{V\rho C_p} \\
\beta_{22} &= \frac{\overline{W}}{V\rho} \\
\beta_{23} &= \frac{\overline{T}_{ri} - \overline{T}_r}{V\rho}
\end{aligned}
$$

Equation (E.4), which describes the enthalpy balance in the jacket, has the following linear form:

$$
\begin{aligned}
M_j C_{pj} \frac{dT_{jo}(t)}{dt} &= UA(T_r(t) - T_{jo}(t)) - W_j(t) C_{pj}(T_{jo}(t) - T_{ji}(t)) \\
\frac{dT_{jo}(t)}{dt} &= \alpha_3 + \alpha_{32} T_r(t) + \alpha_{33} T_{jo}(t) + \beta_{34} T_{ji}(t) + \beta_{35} W_j(t) \qquad \text{(E.15)} \\
\alpha_3 &= \frac{\overline{W}_j(\overline{T}_{jo} - \overline{T}_{ji})}{\overline{M}_j} \\
\alpha_{32} &= \frac{UA}{M_j C_{pj}} \\
\alpha_{33} &= -\left(\frac{\overline{W}_j C_{pj} + UA}{M_j C_{pj}}\right) \\
\beta_{34} &= \frac{\overline{W}_j}{M_j} \\
\beta_{35} &= \frac{\overline{T}_{ji} - \overline{T}_{jo}}{M_j}
\end{aligned}
$$

One can re-write equations (E.13-E.15) and group the constant terms (initial conditions included):

$$
\begin{aligned}
\frac{dC_A(t)}{dt} &= \alpha_{11} C_A(t) + \alpha_{12} T_r(t) + \left\{\alpha_1 + \beta_{11} \overline{C}_{Ai} + \beta_{13} \overline{W}\right\} \\
\frac{dT_r(t)}{dt} &= \alpha_{21} C_A(t) + \alpha_{22} T_r(t) + \alpha_{23} T_{jo}(t) + \left\{\alpha_2 + \beta_{22} \overline{T}_{ri} + \beta_{23} \overline{W}\right\} \\
\frac{dT_{jo}(t)}{dt} &= \alpha_{32} T_r(t) + \alpha_{33} T_{jo}(t) + \left\{\alpha_3 + \beta_{34} \overline{T}_{ji}(t) + \beta_{35} \overline{W}_j\right\}
\end{aligned}
$$

Using matrix notation:

$$
\begin{bmatrix} \frac{dC_A(t)}{dt} \\ \frac{dT_r(t)}{dt} \\ \frac{dT_{jo}(t)}{dt} \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & 0 \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ 0 & \alpha_{32} & \alpha_{33} \end{bmatrix} \begin{bmatrix} C_A(t) \\ T_r(t) \\ T_{jo}(t) \end{bmatrix} + \begin{bmatrix} \alpha_1 + \beta_{11} \overline{C}_{Ai} + \beta_{13} \overline{W} \\ \alpha_2 + \beta_{22} \overline{T}_{ri} + \beta_{23} \overline{W} \\ \alpha_3 + \beta_{34} \overline{T}_{ji}(t) + \beta_{35} \overline{W}_j \end{bmatrix} \qquad \text{(E.16)}
$$

Equation (E.16) can be represented as a *JMLG* model:

$$
\begin{aligned}
x_{t+1} &= A x_t + B w_{t+1} + F u_{t+1} \qquad \text{(E.17)} \\
y_t &= C x_t + D v_t + G u_t \qquad \text{(E.18)}
\end{aligned}
$$

with these definitions:

$$
x_{t+1} = \begin{bmatrix} \frac{dC_A(t)}{dt} \\ \frac{dT_r(t)}{dt} \\ \frac{dT_{jo}(t)}{dt} \end{bmatrix} \quad y_t = \begin{bmatrix} C_A(t) \\ T_r(t) \\ T_{jo}(t) \end{bmatrix} \quad u_t = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{bmatrix}
$$

w_t and v_t are $\mathcal{N}(0, 1)$. The matrices are defined as:

$$
A = \begin{bmatrix} \alpha_{11} & \alpha_{12} & 0 \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ 0 & \alpha_{32} & \alpha_{33} \end{bmatrix} \quad F = \begin{bmatrix} \alpha_1 + \beta_{11} \overline{C}_{Ai} + \beta_{13} \overline{W} & 0 & 0 \\ \alpha_2 + \beta_{22} \overline{T}_{ri} + \beta_{23} \overline{W} & 0 & 0 \\ \alpha_3 + \beta_{34} \overline{T}_{ji}(t) + \beta_{35} \overline{W}_j & 0 & 0 \end{bmatrix}
$$

$B = n_{process}I^{3\times3}$, $C = I^{3\times3}$, $D = n_{measurement}I^{3\times3}$, $G = 0^{3\times3}$. $n_{process}$ and $n_{measurement}$ are the process and measurement noises. One can get the sampled state space representation from the continuous one using standard control engineering theory [Ogata, 1995]. The continuous state space representation for normal operating conditions ($z_t = 1$) is:

$$A(z_t = 1) = \begin{bmatrix} -0.2504 & -0.002182 & 0 \\ 2.413 & -0.05706 & 0.03333 \\ 0 & 0.15 & -0.4125 \end{bmatrix} \quad F(z_t = 1) = \begin{bmatrix} 1.3150 & 0 & 0 \\ -1.8319 & 0 & 0 \\ 21.0 & 0 & 0 \end{bmatrix} \tag{E.19}$$

the matrices $\{ B(z_t = 1), C(z_t = 1), D(z_t = 1), G(z_t = 1) \}$ do not change.

E.1.3 4 Discrete modes

The sampled state space representation for normal operating conditions ($z_t = 1$), using 0.1 min as a sampling rate, is:

$$A(z_t = 1) = \begin{bmatrix} 0.9752 & -0.0002149 & -3.55(10)^{-7} \\ 0.2376 & 0.9943 & 0.003256 \\ 0.001767 & 0.01465 & 0.9596 \end{bmatrix} \quad F(z_t = 1) = \begin{bmatrix} 0.1299 & 0 & 0 \\ -0.1635 & 0 & 0 \\ 2.056 & 0 & 0 \end{bmatrix} \tag{E.20}$$

We changed some operating conditions in order to simulate faulty behaviours. Assuming only U (the heat transfer coefficient) changes, we built 3 new discrete modes ($z_t = 2, 3, 4$) whose sampled state space matrices are:

$$A(z_t = 2) = \begin{bmatrix} 0.9752 & -0.0002144 & -3.36(10)^{-7} \\ 0.2386 & 0.9945 & 0.003095 \\ 0.001686 & 0.01393 & 0.9603 \end{bmatrix} \quad F(z_t = 2) = \begin{bmatrix} 0.1301 & 0 & 0 \\ -0.1662 & 0 & 0 \\ 2.057 & 0 & 0 \end{bmatrix} \tag{E.21}$$

$$A(z_t = 3) = \begin{bmatrix} 0.9751 & -0.000214 & -3.18(0)^{-7} \\ 0.2397 & 0.9946 & 0.002933 \\ 0.001605 & 0.0132 & 0.961 \end{bmatrix} \quad F(z_t = 3) = \begin{bmatrix} 0.1303 & 0 & 0 \\ -0.1689 & 0 & 0 \\ 2.058 & 0 & 0 \end{bmatrix} \tag{E.22}$$

$$A(z_t = 4) = \begin{bmatrix} 0.975 & -0.0002134 & -3(10)^{-7} \\ 0.2408 & 0.9948 & 0.002771 \\ 0.001523 & 0.01247 & 0.9618 \end{bmatrix} \quad F(z_t = 4) = \begin{bmatrix} 0.1305 & 0 & 0 \\ -0.1719 & 0 & 0 \\ 2.058 & 0 & 0 \end{bmatrix} \tag{E.23}$$

$B(z_t) = n_{process}I^{3\times3}$, $C(z_t) = I^{3\times3}$, $D(z_t) = n_{measurement}I^{3\times3}$, and $G(z_t) = 0^{3\times3}$ where $z_t = 2, 3, 4$.
Figure E.1 shows a simulation of the nonlinear system and the JMLG model.

E.1.4 10 Discrete modes

Based on equations (E.20-E.23) we added 6 more discrete modes represented by equations (E.24-E.29). These new discrete modes were described in Chapter 6 section 6.2.2.

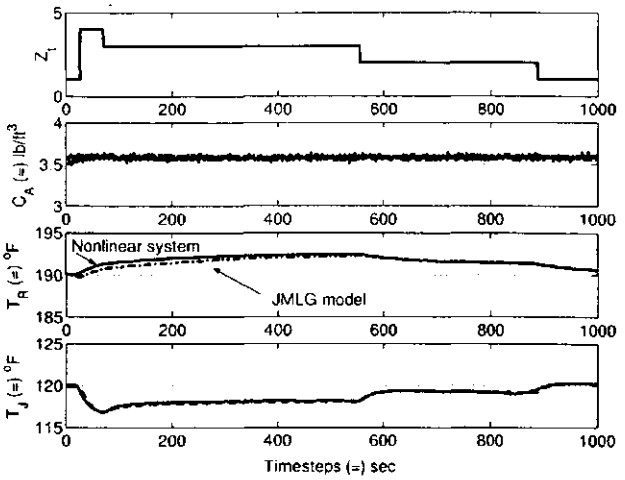

Figure E.1: CSTR (4 discrete modes). Nonlinear and JMLG simulation.

$$A(z_t=5)=\begin{bmatrix} 0.977 & -0.0002 & -3(10)^{-7} \\ 0.2168 & 0.994 & 0.0033 \\ 0.0016 & 0.0147 & 0.9596 \end{bmatrix} \quad F(z_t=5)=\begin{bmatrix} 0.1142 & 0 & 0 \\ -0.03978 & 0 & 0 \\ 2.057 & 0 & 0 \end{bmatrix} \tag{E.24}$$

$$A(z_t=6)=\begin{bmatrix} 0.9787 & -0.0002 & -2(10)^{-7} \\ 0.1959 & 0.9938 & 0.0033 \\ 0.0015 & 0.0147 & 0.9596 \end{bmatrix} \quad F(z_t=6)=\begin{bmatrix} 0.0988 & 0 & 0 \\ 0.0802 & 0 & 0 \\ 2.058 & 0 & 0 \end{bmatrix} \tag{E.25}$$

$$A(z_t=7)=\begin{bmatrix} 0.9752 & -0.0002 & -4(10)^{-7} \\ 0.2382 & 0.9943 & 0.0033 \\ 0.0018 & 0.0147 & 0.9609 \end{bmatrix} \quad F(z_t=7)=\begin{bmatrix} 0.13 & 0 & 0 \\ -0.1653 & 0 & 0 \\ 1.954 & 0 & 0 \end{bmatrix} \tag{E.26}$$

$$A(z_t=8)=\begin{bmatrix} 0.9751 & -0.0002 & -4(10)^{-7} \\ 0.2388 & 0.9943 & 0.0033 \\ 0.0018 & 0.0147 & 0.9621 \end{bmatrix} \quad F(z_t=8)=\begin{bmatrix} 0.1301 & 0 & 0 \\ -0.1668 & 0 & 0 \\ 1.853 & 0 & 0 \end{bmatrix} \tag{E.27}$$

$$A(z_t=9)=\begin{bmatrix} 0.9753 & -0.0002 & -4(10)^{-7} \\ 0.2365 & 0.9943 & 0.0033 \\ 0.0018 & 0.0147 & 0.9596 \end{bmatrix} \quad F(z_t=9)=\begin{bmatrix} 0.1296 & 0 & 0 \\ -0.1608 & 0 & 0 \\ 1.953 & 0 & 0 \end{bmatrix} \tag{E.28}$$

$$A(z_t=10)=\begin{bmatrix} 0.9754 & -0.0002 & -4(10)^{-7} \\ 0.2354 & 0.9943 & 0.0033 \\ 0.0018 & 0.0147 & 0.9596 \end{bmatrix} \quad F(z_t=10)=\begin{bmatrix} 0.1294 & 0 & 0 \\ -0.1584 & 0 & 0 \\ 1.85 & 0 & 0 \end{bmatrix} \tag{E.29}$$

Again, $B(z_t) = n_{process} I^{3\times 3}$, $C(z_t) = I^{3\times 3}$, $D(z_t) = n_{measurement} I^{3\times 3}$, and $G(z_t) = 0^{3\times 3}$ where $z_t = 5, \ldots, 10$.

Figure E.2 shows a simulation of the nonlinear system and the JMLG model (10 discrete modes). The left graphs were generated with less measurement noise than the right ones.

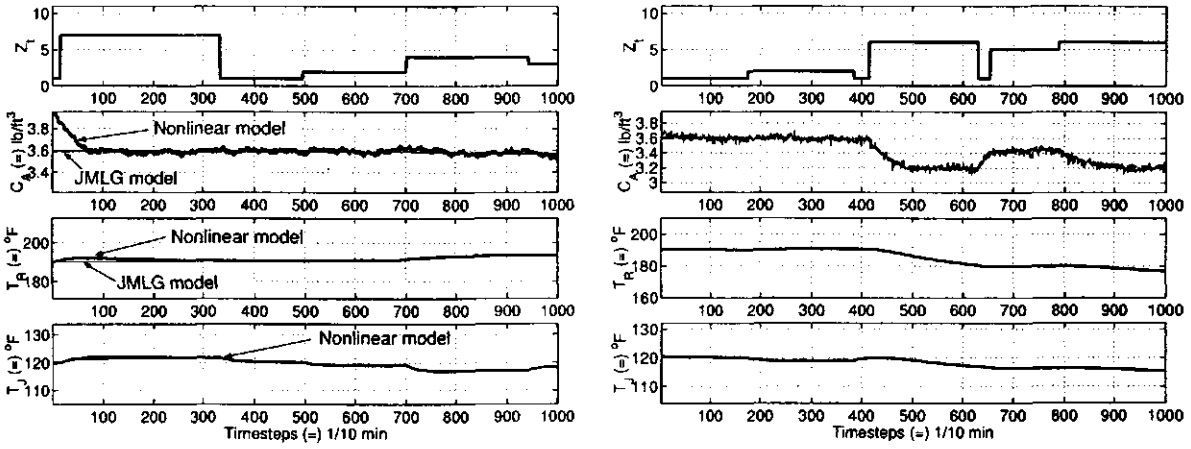

Figure E.2: CSTR (10 discrete modes). Nonlinear and JMLG simulation. The top plots show the discrete mode over time. The remaining plots show the corresponding concentration of A, reactor temperature, and jacket temperature. The non-linear model and the *JMLG* model are compared for each variable. The right system has a higher noise level than the left one.

E.1.5 Results

Figure E.3 shows an example with measurement noise level $n_{measurement} = 6n_{process}$. Diagnosis error versus number of particles and computing time is graphed for each inference algorithm.

E.2 Simulated Mobile Robot

E.2.1 Diagnosis/estimation tests

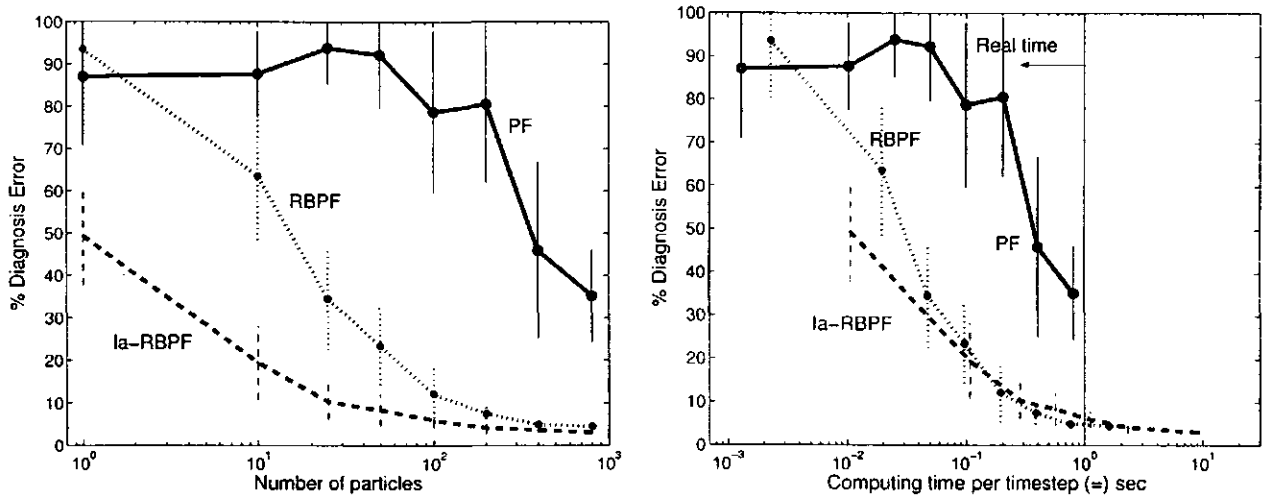

Figure E.3: CSTR (10 discrete modes). Diagnosis error. The left graph shows diagnosis error versus number of particles, while the right graph shows diagnosis error versus computing time per time step. The measurement noise level is high.

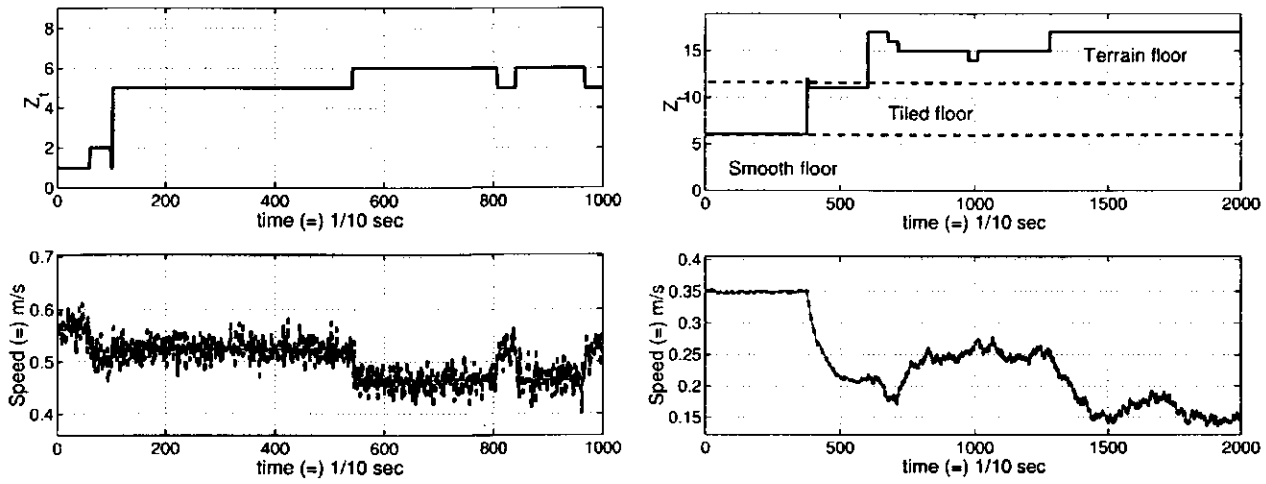

Figure E.4: Mobile robot. Random sequences. The left plot is for 8 discrete modes. The right plot corresponds to 18 discrete modes.

Appendix F

NASA experiments

F.1 RIACS/NASA Ames Research Center

F.1.1 *la-RBPF* in the K-9 planetary rover

A linearized model of the suspension system of the *K-9* planetary rover was applied to detect rocks using real data. This data was generated by driving the *K-9* rover over three different rocks in an outdoor sandbox at NASA Ames. During this experimental test, the rocker and bogey angles were measured. Based on this data, the discrete mode was estimated using both standard *PF* and *la-RBPF*, using 32 and 4 particles respectively. These numbers of particles allow us to get results in real-time. Table F.1 describes the different discrete modes for this domain.

Table F.1: K-9 planetary rovers. Discrete modes description

z_t	Description
1	Flat driving
2	Driving over a rock with the front wheel
3	Driving over a rock with the middle wheel
4	Driving over a rock with the rear wheel
5	There are rocks between the front and the middle wheels
6	There are rocks between the middle and the rear wheels

It is clear from the continuous data (rocker and bogey angles) when the rover is driving over a rock, Figure F.1. The logical discrete mode sequence is 1,2,5,3,6,4 for each of these three rocks (starting at time steps 60, 140 and 210 approximately). The *la-RBPF* algorithm successfully detects the discrete mode sequence for the first and third rocks. However, it makes some diagnosis errors on the second rock; discrete mode 4 is missed. This may be due to the linearized model that was used. The *PF* algorithm almost misses the third rock; and it makes some diagnosis errors when discrete mode 1 is evident.

Several tests using *K-9* planetary rover clearly show that *la-RBPF* outperforms *RBPF* and *PF*, Figure F.2. Figure F.3 shows the mean squared error (MSE) between the continuous variables of the system and the estimation of them (by *JMLG* model). Note the extreme differences between the three Particle Filtering algorithms. With a

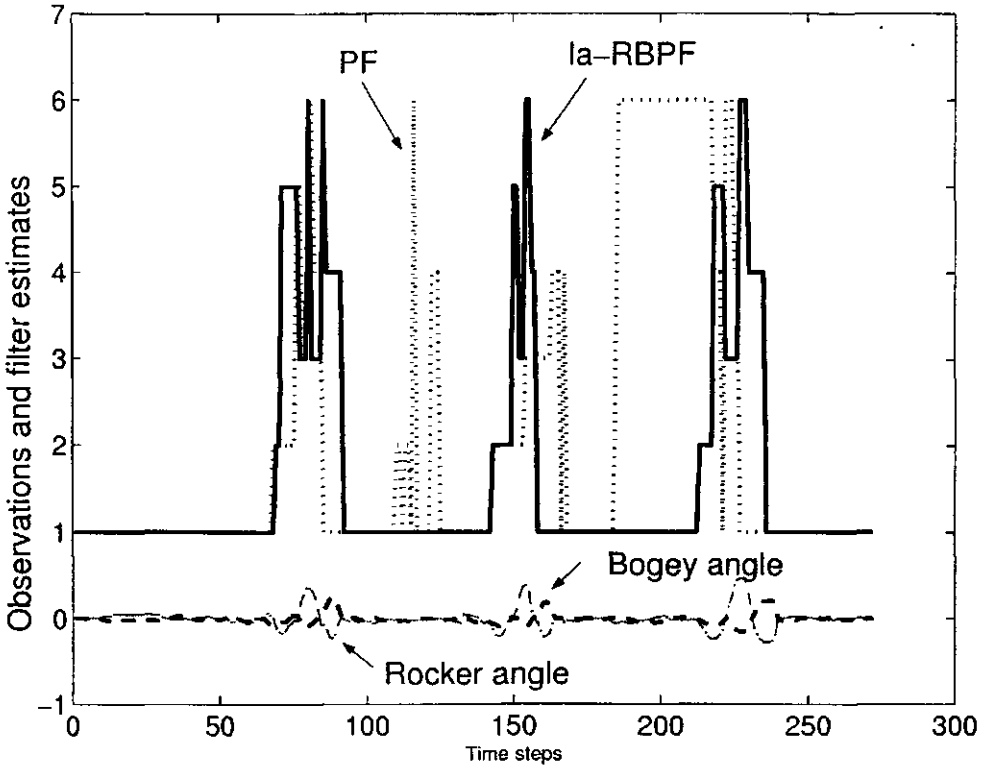

Figure F.1: K-9 rover. Discrete mode estimate on real data. The upper plots show the discrete mode estimates for *PF* and *la-RBPF*, while the lower plots represent the measured rocker and bogey angles when the rovers drive over the three rocks. *Graph provided by Frank Hutter, NASA Ames Research Center.*

MSE of the continuous variables which is between 10^3 and 10^5 times higher than that of *la-RBPF*, it is clearly outperformed.

F.1.2 *la-RBPF* in Marsokhod planetary rover

Diagnosis was performed with the wheel model of the Marsokhod planetary rovers, Figure F.4. Diagnosis is run independently on each of the six wheels. 22 discrete modes are used, 9 of which are normal operational modes, with the others representing faulty conditions. Examples of faulty conditions are: stalled motor, a broken gear, and a broken gear and encoder, see [Washington, 2000; Dearden and Clancy, 2001] for more details. *la-RBPF* was used to diagnose the broken gear and encoder in the right rear wheel [de Freitas *et al.*, 2003]. The rover was set to an idle state and then given the command to start. Figure F.5 plots the number of measurements the algorithms needed to settle on the diagnosis versus the computing time required. The first point in this graph (for each algorithm) corresponds to 1 particle, then $2, 4, 2^k, \ldots, 2^{15}$.

The maximum number of measurements after the start command is issued was 46; if the fault is never diagnosed then 46 is taken by default. Figure F.5 shows that *la-RBPF* can diagnose efficiently using less measurements with less computing time. Only *RBPF* and *la-RBPF* are able to get good results in real-time for this 22 discrete mode domain. Additionally, the *la-RBPF* estimation variance is significantly lower even for fewer particles.

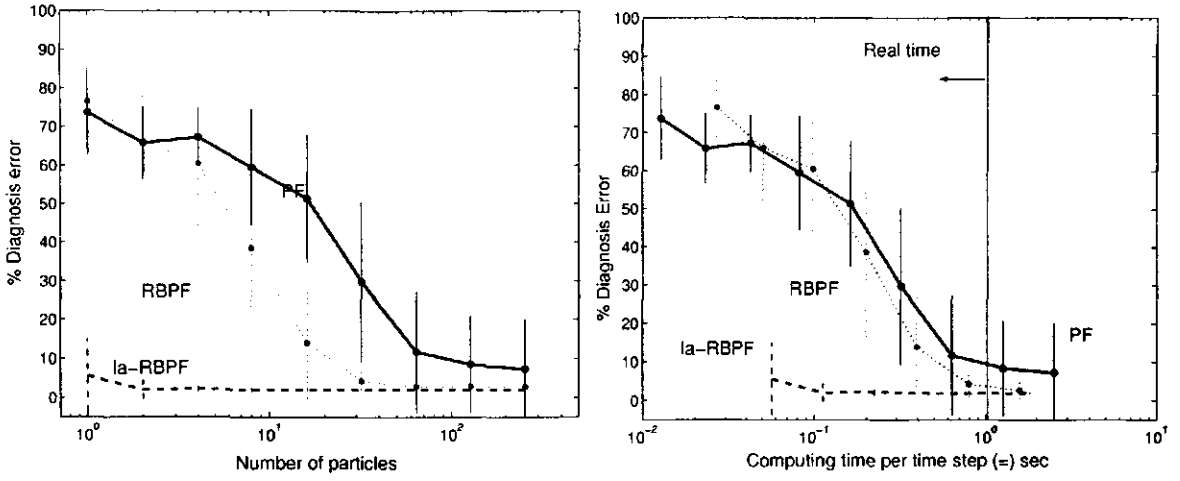

Figure F.2: K-9 rover. Diagnosis error. The left graph shows diagnosis error versus number of particles, while the right graph shows diagnosis error versus computing time per time step. *Graphs provided by Frank Hutter, NASA Ames Research Center.*

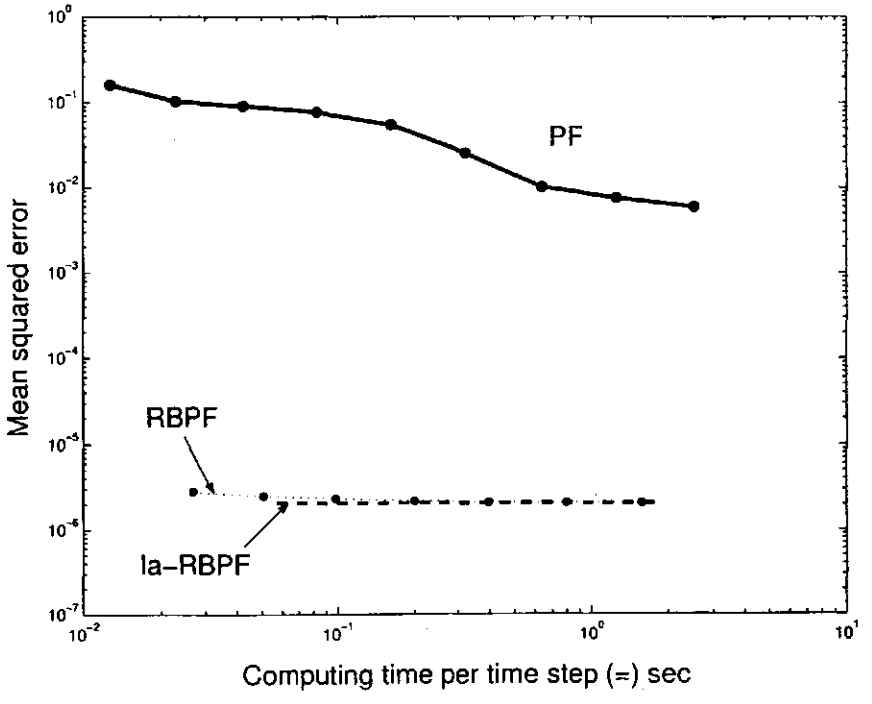

Figure F.3: K-9 rover. Mean squared error. Note the logarithmic scale for both the axes *Graphs provided by Frank Hutter, NASA Ames Research Center.*

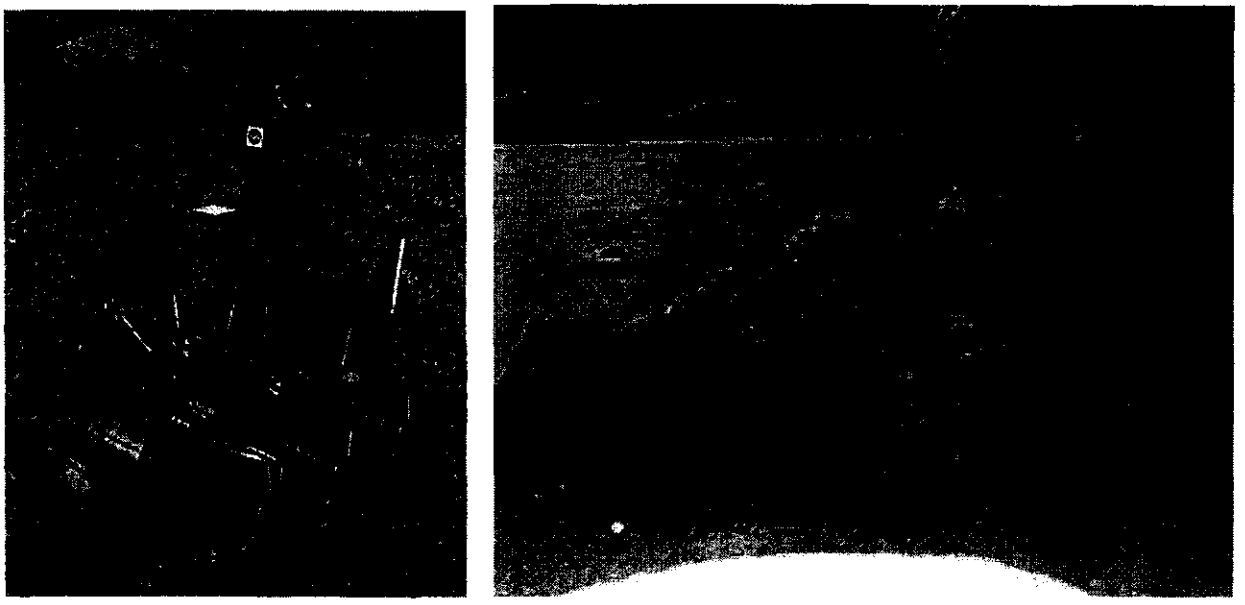

Figure F.4: Marsokhod planetary rovers. Marsokhod is a medium-sized planetary rover built on a Russian chassis. The rover has six independently driven wheels. *Pictures provided by Tom Trower, NASA Ames Research Center.*

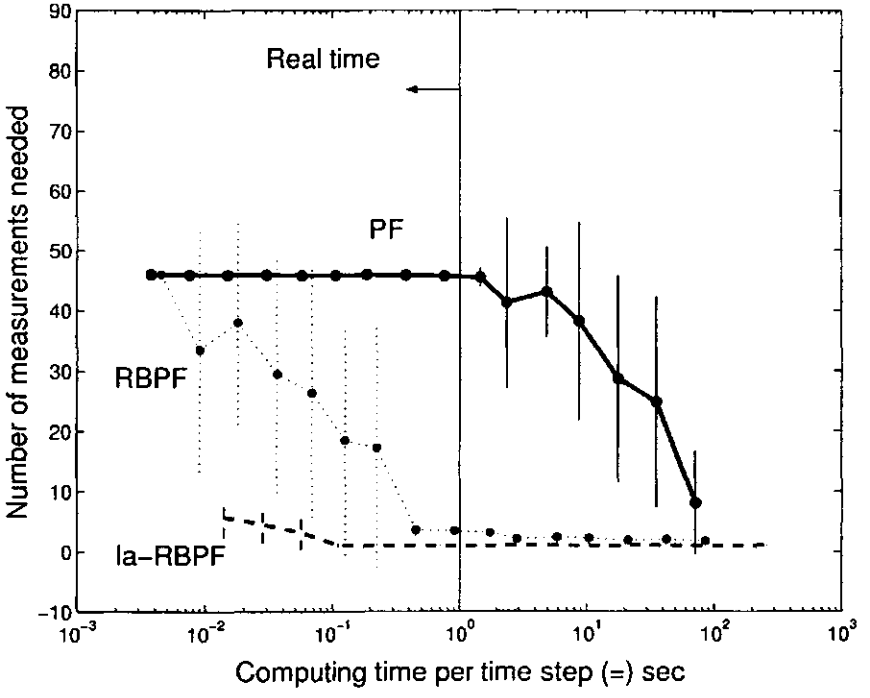

Figure F.5: Marsokhod rover. Number of measurements needed to diagnose a fault. For each algorithm the data points represent the average and standard deviation based on 25 independent runs. *Graph provided by Frank Hutter, NASA Ames Research Center.*