Batch Control and Diagnosis

Olsson, Rasmus

2005

Document Version:
Publisher's PDF, also known as Version of record

Batch Control and Diagnosis

Rasmus Olsson

Automatic Control

Batch Control and Diagnosis

Batch Control and Diagnosis

Rasmus Olsson

Department of Automatic Control
Lund Institute of Technology
Lund, June 2005

Till Min Mormor Britta Hillbom (1919–2004)
—som alltid undrade när jag skulle
skaffa mig ett riktigt arbete.

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 LUND
Sweden

ISSN 0280–5316
ISRN LUTFD2/TFRT--1073--SE

Printed in Sweden by Media-Tryck.
Lund 2005

Abstract

Batch processes are becoming more and more important in the chemical process industry, where they are used in the manufacture of specialty materials, which often are highly profitable. Some examples where batch processes are important are the manufacturing of pharmaceuticals, polymers, and semiconductors.

The focus of this thesis is exception handling and fault detection in batch control. In the first part an internal model approach for exception handling is proposed where each equipment object in the control system is extended with a state-machine based model that is used on-line to structure and implement the safety interlock logic. The thesis treats exception handling both at the unit supervision level and at the recipe level. The goal is to provide a structure, which makes the implementation of exception handling in batch processes easier. The exception handling approach has been implemented in JGrafchart and tested on the batch pilot plant Procel at Universitat Politècnica de Catalunya in Barcelona, Spain.

The second part of the thesis is focused on fault detection in batch processes. A process fault can be any kind of malfunction in a dynamic system or plant, which leads to unacceptable performance such as personnel injuries or bad product quality. Fault detection in dynamic processes is a large area of research where several different categories of methods exist, e.g., model-based and process history-based methods. The finite duration and non-linear behavior of batch processes where the variables change significantly over time and the quality variables are only measured at the end of the batch lead to that the monitoring of batch processes is quite different from the monitoring of continuous processes. A benchmark batch process simulation model is used for comparison of several fault detection methods. A survey of multivariate statistical methods for batch process monitoring is performed and new algorithms for two of the methods are developed. It is also shown that by combining model-based estimation and multivariate methods fault detection can be improved even though the process is not fully observable.

Acknowledgments

I would like to thank my supervisor Karl-Erik Årzén for all the help and stimulating discussions during the work, which has led to this thesis. Karl-Erik is always able to make things work on the fly. If you come to him with a problem it will be solved in real-time. Thank you for the inputs and encouragements during the hard parts of this work!

I would also like to thank all my colleagues at the Department of Automatic Control for making it a pleasure to go to work every day. Special thanks to Henrik Sandberg for numerous discussions and proof reading, Rolf Braun for all the help with hardware and practical problems, Björn Wittenmark for proof reading the thesis, Anders Blomdell and Leif Andersson for the help with my computer and LaTeX, and the Sparta lunch club.

This work has been supported by the Center for Chemical Process Design and Control, CPDC, and the Swedish Foundation for Strategic Research, SSF, which I am very grateful for. The CPDC graduate school has been a good environment for discussions and exchange of ideas between different disciplines in chemical engineering. I would especially like to thank Anders Björk for fruitful discussions and cooperation during the implementation of the MSPC methods.

The cooperation with the Department of Chemical Engineering at Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, has been very fruitful. Thanks to Jordi Canton, Diego Ruiz, Estanislao Musulin and Professor Luis Puigjaner.

All the people outside the world of automatic control, who have contributed to my life in different ways, also deserve some credits. A special thanks goes to my Mother, Father, and Sister for always being there for me. Props go to Cia, the guys at UCPA, Ridea, and LBWS, and all my friends in Skåne, Gbg, Val d'Isère, and the rest of the world.

The development of JGrafchart is funded by LUCAS – Center for Applied Software Research at Lund Institute of Technology and the EC/GROWTH project CHEM aimed at developing advanced decision support systems for different operator-assisted tasks, e.g., process monitoring, data and event analysis, fault detection and isolation, and operations support, within the chemical process industries. The CHEM project is funded by the European Community under the Competitive and Sustainable Growth Programme (1998-2002) under contract G1RD-CT-2001-00466.

Contents

1

Introduction

1.1 Background and Motivation

Batch processes are becoming more and more important in the chemical process industry, where they are used in the manufacture of specialty materials, which often are highly profitable. Some examples where batch processes are important are the manufacturing of pharmaceuticals, polymers, and semiconductors. In continuous processes grade and product changes, as well as start-up and shut-down phases can also be seen as batch processes. Transitions in continuous processes result in production time loss and may result in a product which is not on specification and may have to be reprocessed or destroyed. Techniques developed for batch processes can be used to minimize this.

In a batch process a certain amount of material is transformed, often in several steps, to reach a new state or form through processing in one or several units. The formal definition of a batch process in the batch control standard S88.01 [ANSI/ISA, 1995; IEC, 1997] is:

> "A process that leads to the production of finite quantities of material by subjecting quantities of input material to an ordered set of processing activities over a finite period of time using one or more equipment units."

In this thesis the term batch process means both the production method from raw materials to the final product and the more delimited definition concerning the reaction in a batch reactor.

The manufacturing of products in factories using batch processes is done almost in the same way as we prepare our dinners at home. The manufacturing follows a recipe where one is told to use ingredients and vessels of a specified sort, which temperatures should be used, for how

long, and so on. A recipe for making a batch of pharmaceuticals may look very much the same as the one for baking bread. Of course there are many more restrictions and regulations, which need to be considered when producing pharmaceuticals.

From a control point of view a batch process combines the characteristics of continuous-flow production with those of discrete, part-based production. A batch can be viewed as a discrete entity, which during production moves between different production units. However, during the transition phase from one unit to the next unit and during certain operations within a unit, e.g., fed-batch operations, the batch is more naturally described as a continuous process. The mixed discrete-continuous nature and the recipe-driven production of batch processes make batch control a challenging problem. A batch control system must support a large number of functions in addition to the basic regulatory control. Some examples are production planning, production scheduling, recipe management, resource arbitration and allocation, batch report generation, unit supervision, fault detection and identification, and exception handling. A graphical approach is convenient to illustrate the different steps of batch processes. Both which actions could or should be taken at a certain instant in time and which history of operations led to the current state of the process can be depicted in a clear way.

The focus of this thesis is exception handling and fault detection in batch control. Exception handling in recipe based batch control is a critical element for achieving long-term success in batch production. It is reported to constitute 40-60 percent of the batch control design and implementation effort [Christie, 1998]. Some examples of exception handling situations could be:

- A valve that fails to respond to an `Open` or `Close` command.
- A chemical reaction that does not begin as expected after ingredients have been added.
- The wrong amount of an ingredient has been charged to the unit.
- A recipe that requires a shared resource which is already in use.
- Fouling on heat exchanger surfaces.
- An equipment unit that is not ready for charging when the preceding vessel expects to transfer material.
- An emergency stop that must be performed due to a potentially hazardous situation.

Detection, identification, and correct handling of exceptions of these types is a key element in process safety, consistent product quality, and production cost minimization.

1.2 Research Approach

The thesis has two main parts. In the first part the exception handling strategies in [Olsson, 2002] are further developed. The strategies are implemented and integrated with different toolboxes to form a control system for a batch pilot plant. The implementation uses the batch control standard S88 (IEC 61512) [ISA, 1995; ANSI/ISA, 1995; IEC, 1997] and the BatchML standard [World Batch Forum, 2003] for communication between the different toolboxes. Recently a lot of focus has been put on standardization of the models and terminology used in batch control, e.g., NAMUR [NAMUR, 1992] and S88. However, so far very little has been specified in the area of exception handling. Different discrete-event formalisms have been used in research on the discrete nature of batch control, see, e.g., [Fritz *et al.*, 1999; Hanisch and Fleck, 1996; Tittus and Åkesson, 1999]. However, most of this work has been concerned with scheduling, resource allocation, and simulation. The second part of the thesis is aimed at understanding how historical data from previously successful batches can be combined with models to improve fault detection and diagnosis. The research tries to describe when different methods give best results and how they can complement each other.

The first part of the thesis has been performed within the Grafchart group at the Department of Automatic Control, Lund Institute of Technology. The aim of this group is to develop improved domain-specific graphical programming languages for control applications, in particular applications of a discrete and sequential nature. The aim is to extend the languages that are commonly used for this purposes within automation, especially Grafcet/Sequential Function Charts, with better support for structuring, abstraction, encapsulation, re-use, and user interaction. With these extensions it becomes easier to implement complex applications. It is easier for the programmers and process operators to get an overview of the application and the probability of programming errors decreases. The research approach is complementary to the work on formal verification methods for languages of the above type, e.g., [Brettschmeider *et al.*, 1996; Genrich *et al.*, 1994].

Recipe-based batch control systems is used as the major example domain in the work. The main reason for this is the high complexity of these systems. The research approach is to investigate different language extensions and how these language extensions simplify the development of batch control systems and allow more and more functions in a batch control system to be implemented within the same programming language framework. As far as possible the ambition is to follow the batch control standard S88 or to suggest additions to the standard. In previous work on Grafchart for sequential programming [Johnsson, 1999] the focus has

been batch process recipe representation, recipe handling, and resource arbitration and allocation. Here this work is extended to also include exception handling on both the recipe level and the equipment level. An internal model approach is proposed where each equipment object in the control system is extended with a state-machine based model that is used on-line to structure and implement the safety interlock logic. The exception detection logic associated with a certain state is only active when the state is active. The internal model provides a safety check to ensure that recipe operations are performed in a correct order. An operation is only allowed to be activated if the state of the equipment unit is within a certain set of allowed states associated with the operation. The thesis treats exception handling both at the unit supervision level and at the recipe level. The goal is to provide a structure, which makes the implementation of exception handling in batch process control systems easier.

The work is based on two new language features in Grafchart: MIMO macro steps and step fusion sets. A MIMO macro step is a macro step with multiple input ports and multiple output ports. It can be conveniently used to represent hierarchical states in state-machine based control systems. The functionality is similar to the super-states in Statecharts [Harel, 1987]. Step fusion sets [Jensen and Rozenberg, 1991] provide a way to have multiple graphical representations, or views, of the same step. Using step fusion sets it is possible to separate the exception-handling logic from the normal operation sequences in a way that improves usability. The proposed approach uses the MIMO macro step functionality to implement the state machines and step fusion sets to provide separation between the exception handling logic and the logic for normal, fault-free operation.

The exception handling approach has been implemented and tested in JGrafchart, an implementation of Grafchart in Java. As a test plant Procel, a batch pilot plant, at the Universitat Politècnica de Catalunya (UPC) in Barcelona, Spain, has been used. Procel consists of three tanks with agitators, heaters, and sensors. The tanks are connected in a highly flexible way to be able to run several different types of recipes in the plant.

The second part of the thesis is focused on fault detection in batch processes. A process fault can be any kind of malfunction in a dynamic system or plant, which leads to unacceptable performance such as bad product quality or personnel injuries. Faults may occur either in the sensors, the actuators, the control system, or any other components of the process. The increasing complexity of the control systems in modern plants demands a higher level of fault tolerance and efficient fault detection and isolation is one part to achieve this.

The finite duration and non-linear behavior of batch processes where the variables change significantly over time and the quality variables are

only measured at the end of the batch lead to that the monitoring of batch processes is quite different from the monitoring of continuous processes. Irreversible reactions makes the process uncontrollable. There might also be normal variation between batches, e.g, initial feed variations, ambient temperature, and fouling, which makes the process behave differently between different batches. Normal variations in several parameters can make the process not fully observable, which makes it impossible to use observer based methods. The question is how to determine if the current batch can be classified as a normal batch, which leads to a final product with quality according to specification.

A benchmark batch process simulation model has been used for the comparison of different fault detection methods. The process consists of two consecutive exothermic first order reactions taking place in a jacketed batch tank reactor and it has been used in several papers concerning batch process monitoring.

Fault detection and isolation is a large research area, which includes several different categories, e.g., quantitative model-based and process history-based methods. In quantitative model-based fault detection and isolation explicit mathematical models are used for the generation of the residuals. The models are either developed by a first-principles or a black-box model is identified from experiments. Most of the methods use linear discrete time black-box models, such as input-output and state-space models, due to the fact that first-principles models are complex and that chemical processes are often non-linear, especially batch processes. A first-principles model is based on physical understanding of the process. Models of chemical processes are usually based on heat balance equations and mass balances with low-order reaction kinetics. In a first-principles model the parameter have physical meaning while in a black-box model this is often not the case. Model-based diagnosis is based on forming residuals where the outputs from the model are compared with the measurements from the plant. Discrepancies between these are indications of faults.

In most cases batch production is repetitive in nature. Information from previous batches may therefore be used to improve control and monitoring of the current and future batches. This structure is used in multivariate statistical methods, which fall under the category of process history-based methods, for batch process monitoring. Multivariate statistical methods use only historical data from successful batch runs to model the normal behavior of the process. Historical data collected from a batch process is three dimensional by nature with the dimensions *Batch*, *Variable*, and *Time*. The data is usually stored in a three-dimensional array $\underline{X} \in \mathbb{R}^{I \times J \times K}$, see Fig. 1.1, where I is the number of batches, J is the number of variables, and K is the number of samples in time. Statistical control limits from the normal batches are derived and future batches are

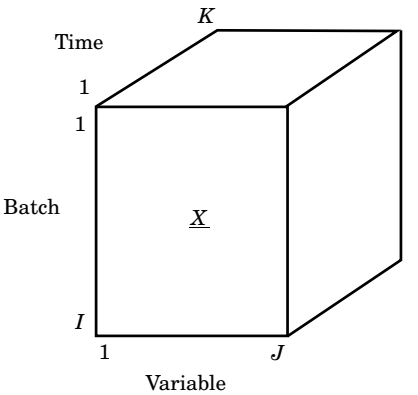

Figure 1.1 Historical data collected from a batch process generates a three dimensional array $\underline{X} \in \mathbb{R}^{I \times J \times K}$.

compared to these limits to classify the batches as normal or abnormal. There exist a number of alternative approaches of multivariate statistical methods for process monitoring, e.g., methods based on principal component analysis, PCA, and methods based on partial least squares techniques, PLS. The focus in this thesis is PCA-based methods.

Model-based diagnosis and history-based diagnosis are quite different in nature, each with its advantages and disadvantages. One aim of this thesis is to compare and evaluate the two approaches against each other. This has been done by applying them to the same simulated benchmark process. The model-based approach chosen is the diagnostic model processor, DMP, method [Petti, 1992]. The reason for this choice is the very general nature of DMP. In the DMP framework it is possible to incorporate model-based techniques of very different kinds, e.g., observer-based methods and methods based on parameter identification. The multivariate methods evaluated include multi-way principal component analysis, MPCA [Wold *et al.*, 1987], using different ways of unfolding the data in $\underline{X}$ [Nomikos and MacGregor, 1994; Wold *et al.*, 1998], batch dynamic PCA (BDPCA) [Chen and Liu, 2002], multi model MPCA, and moving window PCA [Lopes and Menezes, 1998; Lennox *et al.*, 2001b]. It is shown how these different methods are related and when they are equivalent.

An important question is whether it is possible to combine model-based and history-based diagnosis methods, and through this obtain something that has better detection and diagnosis performance than either of the approaches alone. In the thesis an approach is proposed that uses a dynamic model to derive additional variables, or estimates, which then are

fed to the multivariate PCA methods in addition to the plant outputs and inputs. The idea being that adding these estimates should enable the system to detect faults earlier than by using only the original variables. One problem, with normal variations in several parameters, is that even a very simple batch process is not fully observable, which has the consequence that classical estimation techniques, such as extended Kalman filters and recursive least squares, do not give the true estimates. It is shown that combining model-based estimation methods and historical data using multivariate statistical methods can improve the detection of faulty batches even though the estimation of parameters and variables does not give the true values.

Contributions

The main contributions presented in this thesis are:

- A new approach to equipment supervision in recipe-based batch control systems is proposed. The approach uses hierarchical state machines represented in JGrafchart to model the equipment status and the status of the procedures executing in the equipment.
- A new approach for representing exception-handling logic at the recipe-level is proposed. The approach gives a clear separation between exception handling logic and the logic for normal operation.
- Different possibilities for combining the two above approaches is suggested and implemented in JGrafchart. The combined approaches are experimentally verified on a realistic batch pilot plant.
- An extensive survey of methods available for batch process monitoring using multivariate statistical methods is performed. The performance of the methods is evaluated using a benchmark simulated batch process.
- New algorithms are proposed for the multivariate statistical methods BDPCA and DPARAFAC for batch process monitoring.
- A new way of combining model-based estimation and multivariate statistic methods is proposed to improve fault detection of batch processes, when normal batch-to-batch variation is present.
- The methods Diagnostic Model Processor and Deep Model Algorithm are applied to a benchmark simulated batch process.

Publications

The thesis is based on the following publications:

- Olsson, R. and K-E. Årzén: "Exception Handling in Recipe-Based Batch Control". *Proc. of ADPM2000 The 4th International Conference on Automation of Mixed Processes, Dortmund, Germany, 2000*.
- Olsson, R., H. Sandberg and K-E. Årzén: "Development of a Batch Reactor Laboratory Process", *Reglermötet 2002, Linköping, Sweden, 2002*.
- Olsson R. and K-E. Årzén: "Exception Handling in S88 using Grafchart". *Proc. of World Batch Forum North American Conference 2002, Woodcliff Lake, NJ, USA, 2002*.
- Årzén, K-E., R. Olsson and J. Åkesson: "Grafchart for Procedural Operator Support Tasks". *Proc. of the 15th IFAC World Congress, Barcelona, Spain, 2002*.
- Olsson, R.: "Exception Handling in Recipe-Based Batch Control". *Licentiate thesis, Department of Automatic Control, Lund Institute of Technology, Sweden, 2002*.
- Olsson R. and K-E. Årzén: "A Modular Batch Laboratory Process". *International Symposium on Advanced Control of Chemical Processes ADCHEM 2003, Hong Kong, Jan, 2004*.
- Musulin, E., M. J. Arbiza , A. Bonfill, L. Puigjaner, R. Olsson, and K-E. Årzén: "Closing the Information Loop in Recipe-Based Batch Production". *15th European Symposium on Computer Aided Process Engineering (ESCAPE 15), Barcelona, Spain, 2005*.

1.3 Outline of the Thesis

In Chapter 2 batch control in general is discussed and parts of the batch control standard S88 are described. Chapter 3 describes the previous work on Grafchart for batch control and the state-machine based approach to exception handling in recipe-based batch control. The topic of Chapter 4 is the testing and verification of the exception handling approach on the Procel pilot plant. An introduction to existing fault detection and diagnosis methods is found in Chapter 5. In Chapter 6 several multivariate statistical methods for fault detection are described. The benchmark simulated batch process is described in Chapter 7. The use of the diagnostic model processor method for batch process monitoring is presented in Chapter 8. A comparison of the multivariate statistical methods developed for batch

processes is given in Chapter 9. Chapter 10 contains a discussion and new results on how to combine model-based estimation and multivariate statistical methods. The last chapter of the thesis, Chapter 11, gives conclusions and suggestions for future research.

2

Batch Control

2.1 Introduction and Motivation

The control of a batch process is quite different from the control of a continuous process. A continuous process usually operates at a certain operating point and the control system tries to keep the process at that point despite disturbances acting on the process. A batch process involves both continuous and sequential parts. For example, when filling a tank, see Fig. 2.1, to a certain level the flow to the tank is continuous. When the desired level is reached the flow is turned off and the next sequential control part can take place, e.g., heating of the tank. The mixed discrete-continuous nature and recipe-driven production of batch processes make batch control a challenging problem. A batch control system must support a large number of functions in addition to the basic regulatory control. The batches need to be scheduled to meet the production plan, recipes should be developed and maintained, shared resources need to be allocated during processing, production reports of batches should be generated and stored, and faults and exceptions need to be detected and taken care of.

Why use automatic control for batch processes? On the surface it seems easy to control the sequential part of batch processes manually. The task is just to fill a tank and then start to heat it as in the example above. The procedure is just to wait until a certain goal is reached and then start the next part. However, once there is more than one option of which unit to use, there are common resources to be shared between different units, or operations need to be synchronized, the task becomes unmanageable. One of the benefits of automatic control is that the routine operations become fewer for the operators so they can concentrate on more important issues, e.g., exception handling and optimization of the process. Controlling a batch process also leads to repeatability (i.e. consistent product quality

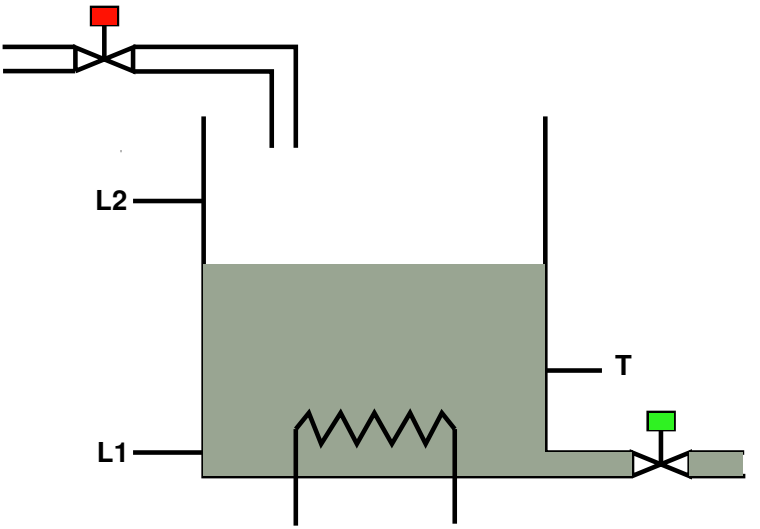

Figure 2.1 Single tank.

between batches) and flexibility (i.e. manufacturing of different products and different grades). Other reasons are the properties of feedback as such and its capabilities of handling disturbances and uncertainties in the process.

The most complex batch process structure is a network-structured, multi-product plant, see Fig. 2.2. In these kind of plants it is possible to produce multiple products at the same time using a finite set of equipment units. The units are organized in a network structure with a high degree of connectivity. The information about the sequential ordering of the operations, the equipment requirements for the different operations, and the product specific parameters for the manufacturing of a certain product are captured in a *recipe*. To execute an operation in an equipment unit, the batch control system must allocate the resource that the equipment unit constitutes. A control system for recipe-based batch production must, hence, include substantially more functions than what is needed in a control system for continuous production. There are a number of books on the subject of control of batch processes, e.g., [Fisher, 1990] and [Rosenhof and Ghosh, 1987].

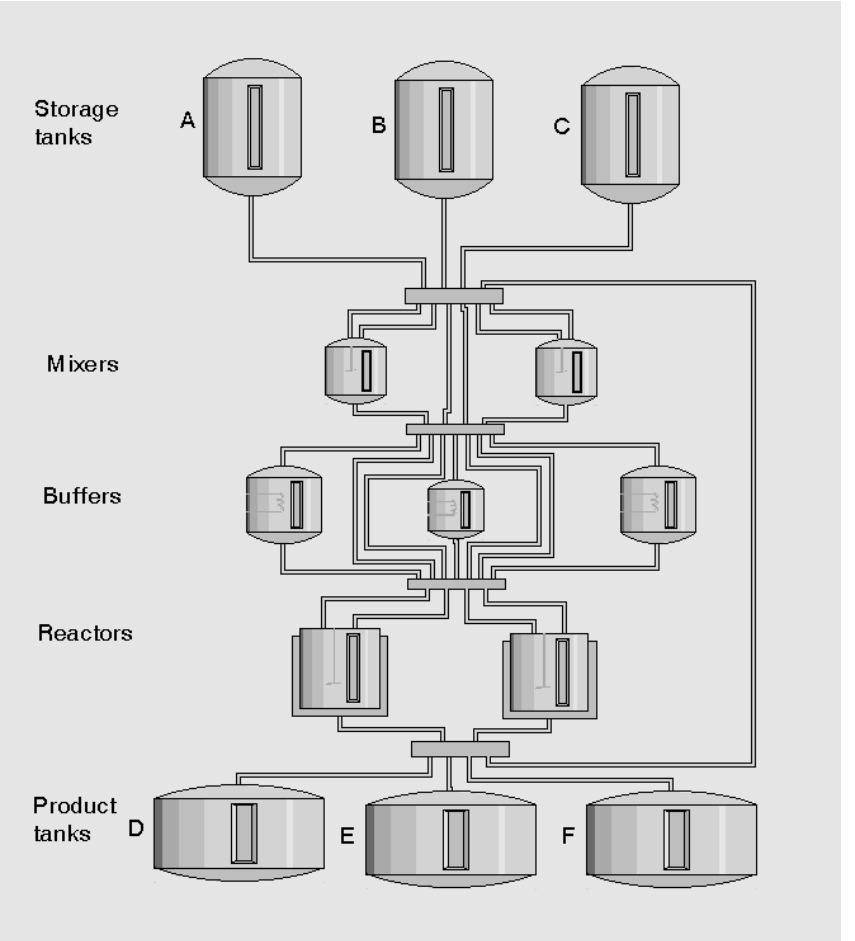

Figure 2.2 Scheme of a network-structured, multi-product plant.

2.2 S88 - Batch Control Standard

During the last decade there have been several initiatives with the aim to standardize batch control systems. The most successful of these attempts is the S88 batch control standard initiated by ISA [ISA, 1995]. The standard is divided into three parts: Part 1 [ANSI/ISA, 1995] deals with models, terminology, and functionality, Part 2 [ANSI/ISA, 2001] deals with data structures and language guidelines, and Part 3 [ANSI/ISA, 2003] deals with general and site recipe models and representation guidelines.

S88 is also known under the committee name SP88 and is equivalent to the international standard IEC 61512 [IEC, 1997; IEC, 2001a]. A book that from the user's perspective describes how to implement S88 for an ice-cream factory is [Parshall and Lamb, 2000].

The S88 standard describes batch control from two different viewpoints: the process view and the equipment view. The process view corresponds to the view of the chemists, and is modeled by the *process model*. The process model is hierarchically decomposed into the following layers: process, process stage, process operation, and process action. The equipment view corresponds to the view of the production personnel and is represented by the *physical model*. The physical model is also a four-layer hierarchical model containing the following four layers: process cell, unit, equipment module, and control module. A process cell contains one or several units. A unit performs one or several major processing activities. The unit consists of equipment modules and/or control modules. An equipment module carries out a finite number of minor processing activities, i.e. phases (described below). In addition to being a part of a unit, an equipment module can also be a stand-alone, shared, or exclusive usage entity within a process cell. The control module, finally, implements a basic control function, e.g., a PID-controller or the logic for handling a valve battery.

The process model and the physical model are linked together by recipes and equipment control. S88 specifies four different recipe types: general, site, master, and control recipes. In this thesis only master and control recipes are considered. Both types of recipes contain four types of information: administrative information, formula information, requirements on the equipment needed, and the procedure that defines how a batch should be produced. The master recipe is targeted towards a specific process cell. It includes almost all information necessary to produce a batch. Each batch is represented by a control recipe. It can be viewed as an instance or copy of the master recipe, that has been completed and/or modified with scheduling, operational, and equipment information.

The procedure in a recipe is structured according to the *procedural model* in S88. The model is hierarchical with four layers: procedure, unit procedure, operation, and phase. A procedure is decomposed into unit procedures, i.e., sets of related operations that are performed within the same unit. The unit procedures are decomposed into operations, which in turn are decomposed into phases. The phase is the smallest element that can perform a process-oriented task, e.g. open a valve or set an alarm limit.

The procedural model on the recipe level is mirrored by the same model on the equipment control level, see Fig. 2.3. The dashed levels could either be contained in the recipe or in the equipment control. The linkage

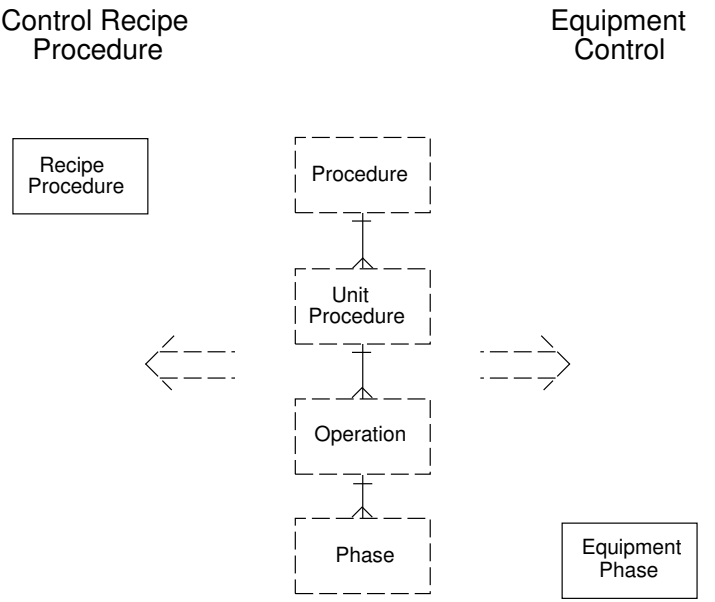

Figure 2.3 Control recipe/Equipment control separation.

between an element in the recipe procedure (unit procedure, operation, or phase) and the corresponding element in the equipment control (equipment unit procedure, equipment operation, or equipment phase) can be viewed as a method call to the equipment unit object that has been allocated to execute the specific recipe procedure element. S88 offers great flexibility concerning at which level the linkage should take place. It is also possible to collapse one or several levels.

Exception Handling in S88

Surprisingly enough S88 mentions very little about exception handling and how this should be integrated with recipes and equipment control. The only thing that is briefly discussed is that *modes* and *states* may be affected.

Equipment entities and procedural elements may have modes, which determine how they respond to commands and how they operate. Procedural elements have three modes: automatic, semi-automatic, and manual, and equipment entities have two modes: automatic and manual. For procedural elements, in the automatic mode the transitions take place as soon as the transition conditions are fulfilled. In semi-automatic mode, the procedure requires manual approval to proceed after the conditions

are fulfilled, and in manual mode the execution of the procedural elements is determined manually. One important issue is how mode changes will propagate. An open question is, e.g., if a unit procedure goes to manual mode should all the lower-level procedural elements within the unit also go to manual mode? The standard does not specify propagation rules. Propagation may be from higher to lower level or vice versa.

Procedural elements and equipment entities may also have states. As an example the following twelve procedural states are mentioned: *idle, running, complete, pausing, paused, holding, held, restarting, stopping, stopped, aborting*, and *aborted*, see Fig. 2.4. If a procedure is in the *idle* state, i.e., it is available, it may be started, and the state will change to *running*. From the *running* state the procedure may be stopped, aborted, held, and paused or the procedure will reach its end and return to the *idle* state through the *complete* state. The state of, e.g., a valve can be: *open, closed*, and *error*. The *error* state would be reached if the valve fails to respond to an open or close command from the control system.

In S88 an exception is defined as an event that occurs outside the normal or desired behavior of batch control. A few examples of events that might need exception handling are stated: control equipment malfunction, fire or chemical spills, or unavailability of raw materials or plant equipment. Exception handling may occur at all levels in the control system. However, how this should or could be done is not mentioned. The exception handling can be incorporated in any part of the control system.

Although there are no specific safety standards for batch processes, general process industry safety standards, such as IEC 61508 [IEC, 1998] and IEC 61511 [IEC, 2003], also apply to batch processes. A Safety Instrumented System[1] (SIS) [ANSI/ISA, 2004] of a batch process need a number of additional functionalities, which are different from an SIS of a continuous process. For more information on process safety issues specific to batch reaction systems see [American Institute of Chemical Engineers – AIChE, 1999].

[1]The instrumentation, controls, and interlocks provided for safe operation of the system.

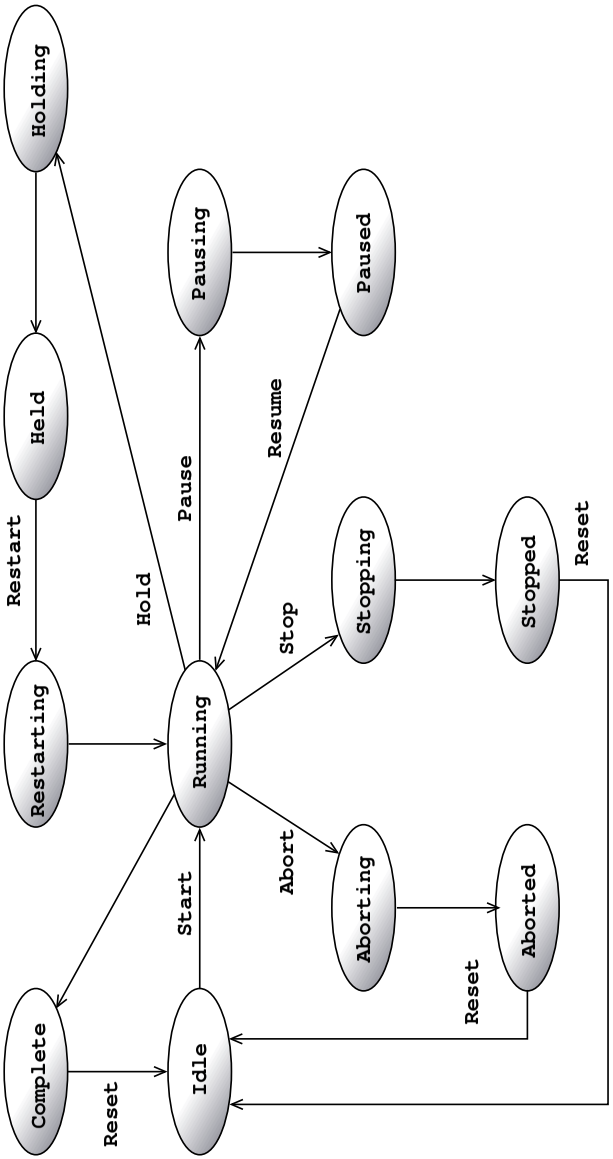

Figure 2.4 The states of a procedural element as described in S88

3

Exception Handling in Recipe-Based Batch Production

3.1 Introduction

An exception is an event that occurs outside the normal or desired behavior of the process. Exception handling is a critical element for achieving long-term success in batch production. It is reported to constitute 40–60 percent of the batch control design and implementation effort [Christie, 1998]. Correct handling of exceptions is a key element in process safety, consistent product quality, and production cost minimization. In short, there is money to be made with well structured and automated exception handling.

In this chapter the work on Grafchart for batch process recipe handling and resource allocation described in [Johnsson and Årzén, 1998a; Johnsson and Årzén, 1998b; Johnsson, 1999], is extended to also include exception handling. An internal model approach is proposed where each equipment unit object is extended with a state machine-based model that is used on-line to structure and implement the safety interlock logic, and to provide a safety check to ensure that recipe operations are performed in a correct order. The goal of this work has been to make the exception handling easier to design and maintain.

As mentioned is Section 2.2 there is no specific safety standard for batch processes, although a number of problems arise especially for these kinds of processes. Some of these problems are stated in [van Beurden and Amkreutz, 2002]:

- Separation between the basic process control system (BPCS) and the safety instrumented system (SIS)
- Synchronization of the process steps between the BPCS and the SIS
- Operator interaction
- Implementation of variable (recipe dependent) alarm levels
- Frequent operational state changes
- Frequent recipe changes

How these kind of problems can be solved using Grafchart is discussed in this chapter. In Section 3.2 an overview is given of Grafchart and how it can be used for recipe handling. This is the basis for the proposed exception handling scheme. Exception handling at the equipment unit level is described in Section 3.3. The corresponding exception handling at the recipe level is described Section 3.4. Section 3.5 discusses the problem with synchronization between units, e.g., in connection with transfers of batches and, finally, Section 3.6 summarizes the proposed schemes.

3.2 Grafchart and Batch Control

Grafchart is a graphical programming language for sequential control applications. It is based on Grafcet, or Sequential Function Charts (SFC), together with ideas from Petri nets, Statecharts [Harel, 1987], high-level programming languages, and object-oriented programming. A thorough presentation of Grafcet and Petri Nets is given in [David and Alla, 1992]. Grafcet/SFC is one of the languages specified for PLC (Programmable Logic Controller) programming in the standard IEC 61131-3 [IEC, 1993], and it is widely used as the representation format for the sequential part of supervisory control. Grafchart has been developed at the Department of Automatic Control at Lund Institute of Technology since 1991 [Årzén, 1991; Årzén, 1993; Årzén, 1994].

JGrafchart is an implementation of Grafchart written in Java and Swing [Sun Microsystems, Inc, 2005]. JGrafchart consists of an integrated graphical editor and run-time system. In the graphical editor the user creates Grafchart function charts by copying language elements from a palette using drag-and-drop. The language elements are placed on a workspace and connected together graphically. After compilation the function charts are executed by the runtime system within the JGrafchart editor. Storage to file is provided in the form of XML using the Java API for XML Processing (JAXP) [Sun Microsystems, Inc, 2002]. For the complete description of JGrafchart, see the home page:
`http://www.control.lth.se/~grafchart`.

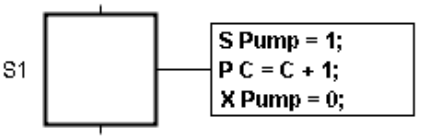

Figure 3.1 Step with actions

The main language elements are steps, transitions, macro steps, procedures, procedure steps, and process steps. A step represents a state where actions are performed. Five types of actions are supported. Stored actions are executed when the step becomes active. The syntax for stored actions is:

```
S "variable" = "expression";
```

Exit actions are executed immediately before a step becomes deactivated. The syntax is:

```
X "variable" = "expression";
```

Periodic actions are executed periodically while the step is active. The rate depends on the rate of the execution thread. The syntax is:

```
P "variable" = "expression";
```

Abort actions are executed when the execution of a step is aborted due to the firing of an exception transition. The syntax is:

```
A "variable" = "expression";
```

Finally, normal actions are used to associate the truth value of a boolean variable with the activation status of the corresponding step. The syntax is the following:

```
N "boolean_variable";
```

The variables can be references to internal variables or to I/O variables. The expressions may contain standard boolean and numerical operators. A step together with some step actions are shown in Fig. 3.1.

Transitions contain a boolean condition and/or event expression that decides when the transition should be fired. The grammar for transition expressions and step actions is defined formally using the Java parser generator JavaCC [Sun Microsystems, 2004]. A parser generator is a tool that reads a grammar specification and converts it to a Java program that can recognize matches to the grammar.

The following special notation can be used for transitions: `step.t` returns the number of scan cycles since the step last was activated, `step.s` returns the time in seconds since the step last was activated, `step.x` returns true if the step is currently active, and `/BoolVar` returns true if the current value of the boolean variable `BoolVar` is true and the previous value was false, i.e., it represents a raising edge event. Falling edge events are handled in a corresponding way.

Grafchart contains three hierarchical abstractions: *macro steps, procedures,* and *workspace objects*. Macro steps are used to represent steps that have an internal structure. The internal structure of the macro step is encapsulated within the macro step. A new feature of the macro step is that it may have more than one enter and exit port.

To re-use sequences in a function chart, a sequence can be placed on the sub-workspace of a procedure. Procedures can be stand-alone entities or methods of objects. For example, an object representing a batch reactor could have methods for charging, discharging, agitating, heating, *etc*. A method is called through a procedure step. The method that will be called is determined by an object reference and a method reference. The procedure is active as long as the procedure step is active. A process step is similar to a procedure step. The difference is that the procedure is started as a separate execution thread and will continue to execute until it reaches its exit step even though the process step has become inactive.

To easier organize programs one can use workspace objects. A workspace object contain a sub-workspace that can be used in the same way as a top-level workspace, i.e., place language elements on to form function charts. Using workspace objects it is possible to create complex variables, e.g., structs.

An open problem in Grafcet is how the logic for the normal operating sequence best should be separated from the exception detection and exception handling logic and sequences. Grafchart contains a number of assisting features for this. An *exception transition* is a special type of transition that may only be connected to macro steps and procedure steps. An ordinary transition connected after a macro step will not become active until the execution has reached the last step of the macro step. An exception transition is active all the time when the macro step is active. If the exception transition condition becomes true while the corresponding macro step is executing the execution will be aborted, abortive actions, if any, are executed, and the step following the exception transition will become active. Macro steps “remember” their execution state from the time they were aborted and it is possible to resume them from that state using a special enter port. Exception transitions have proved to be very useful when implementing exception handling.

Using *connection posts*, it is possible to break a graphical connection

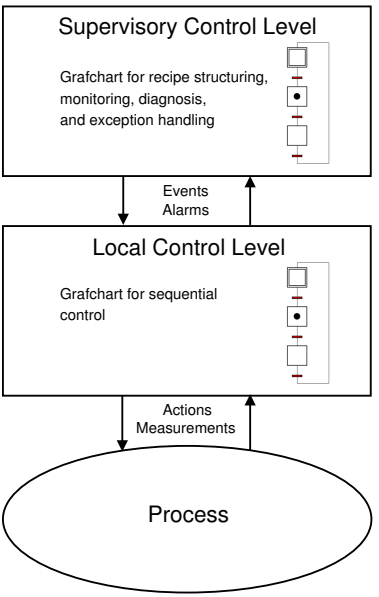

Figure 3.2 Supervision of a sequential process

between, e.g., a step and a transition. In this way it is possible to separate a large function chart into several smaller parts that may be stored on different workspaces. This enhances the readability of the chart. The connection post can be used to separate the normal operating sequence from the exception detection and exception handling logic and sequences.

A new language element to separate function charts is the *step fusion set*. Steps in a step fusion set represents different views of the same step. All the steps in a step fusion set become active when one of the steps becomes active.

Grafchart and Recipe Handling

Grafchart has been used for batch control, recipe handling, and resource allocation, e.g., [Johnsson and Årzén, 1998a; Johnsson and Årzén, 1998b; Johnsson, 1999]. In the work different possibilities for representing recipes and combining recipe execution with resource allocation have been explored. Grafchart can be used at all levels in the hierarchical procedure model, from the programmable logic controller (PLC) level sequence control to the representation of recipes. Grafchart makes it possible to use the same language both at the local control level and at the supervisory control level, see Fig. 3.2.

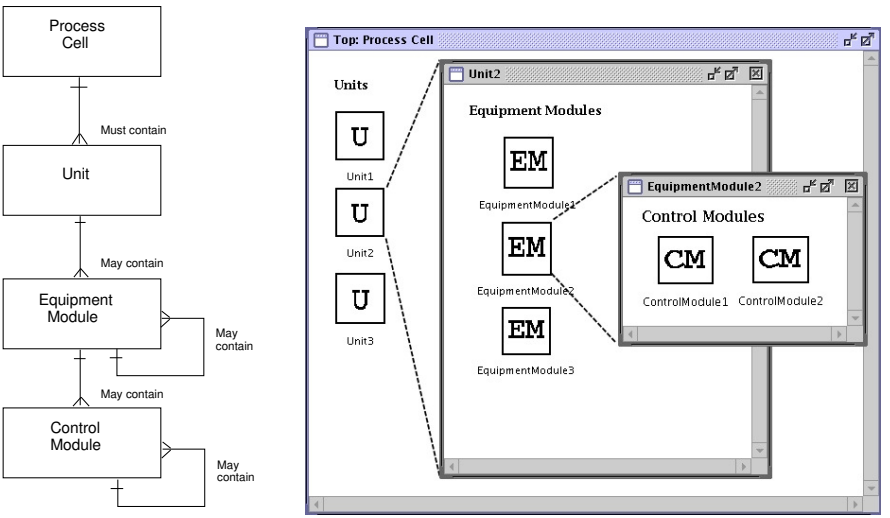

Figure 3.3 The physical model in S88 (left) and in JGrafchart (right)

Physical Model

The physical model in S88 describing the hierarchical relationships between the physical objects involved in batch control can be modeled using workspace objects in JGrafchart. The structure is shown in Fig. 3.3. Each unit is represented by a workspace object. The sub-workspace of a unit contains the equipment modules associated with the unit. Also the equipment modules are represented by workspace objects, which internally contain the associated control modules.

Procedural Model

The hierarchical structure of the S88 procedural model is straightforward to model in Grafchart using macro steps, see Fig. 3.4. The recipe procedure is represented as a function chart. Each unit procedure is represented by a macro step that contains the operations of the unit procedure. The operations are also modeled as macro steps that internally contain phases, also these represented by macro steps. Finally, the phase macro steps contain ordinary steps and transitions modeling the phase logic.

Recipes

Recipes can be represented by procedures or workspace objects in JGrafchart. The procedures can be called from procedure steps or process steps

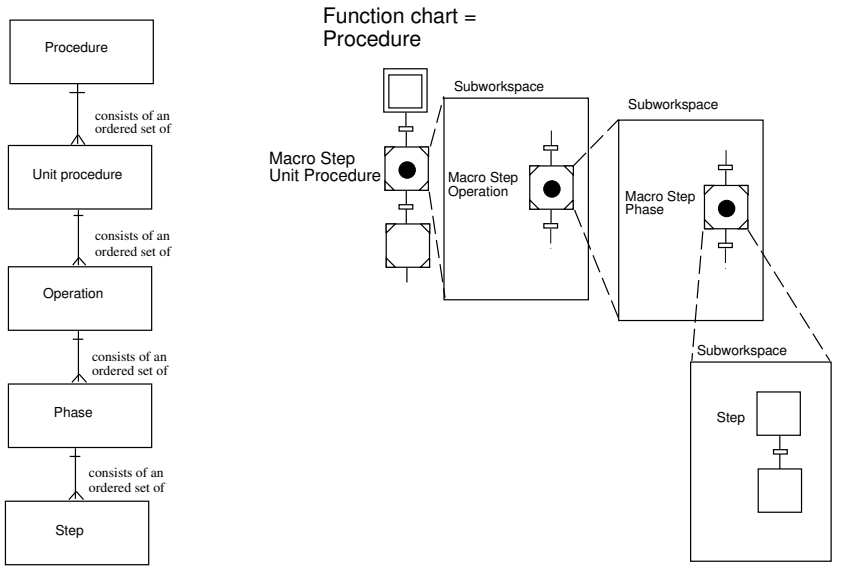

Figure 3.4 S88 Procedural Model (left) and its representation in Grafchart (right).

and workspace objects can be copied to become control recipes after completing the recipe with specific parameters.

The linking between the control recipe and the equipment control is implemented using methods and message passing according to Fig. 3.5. The element in the control recipe where the linking should take place is represented by a procedure step. Depending on at which level the linking takes place, the procedure step could represent a recipe procedure, recipe unit procedure, recipe operation or recipe phase. The procedure step calls the corresponding equipment control element, which is stored as a method in the corresponding equipment object. A number of different ways to represent recipes in Grafchart were proposed in [Johnsson, 1999].

3.3 Unit Supervision

The proposed method for unit supervision is based on augmenting each equipment object (unit, equipment module, control module, *etc*) with a finite state machine as shown for a reactor unit in Fig. 3.6. The reactor unit contains three parts: a set of attributes, Grafchart procedures, and an *equipment-state machine*.

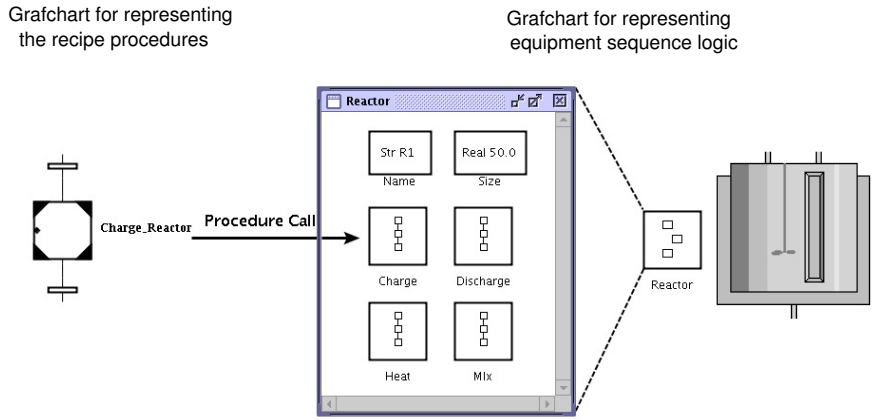

Figure 3.5 Control Recipe/Equipment Control linking.

The attributes could either be attributes of simple types, e.g., max-capacity, or they could be objects, e.g., representing the equipment/control modules within the equipment unit. In the latter case the proposed structure applies recursively, i.e., the equipment/control modules also contain the same three parts as the unit. The Grafchart procedures represent equipment unit operations or phases for, e.g., heating, charging, and mixing as in Fig. 3.5. The equipment-state machine is used to model the behavior of a physical object, i.e., there are states for normal operation and there are states for faults. See Fig. 3.7 for the equipment-state machine of a valve, with the states `Opening`, `Open_OK`, `Closing`, `Closed_OK`, `Error_Open`, and `Error_Closed`.

The equipment-state machine could either be a single automaton modeling, e.g., the behavior of a unit and all its equipment modules, or consist of several smaller parallel automata describing each of the equipment modules in the unit. If the parallel automata are composed they will form the single automaton, see Fig. 3.8. Several smaller parallel automata are probably easier to overview and more user-friendly. This is the approach used in the rest of this chapter. Hierarchical state machines, where each state can contain a whole state machine recursively, can be used to get a better overview of the model.

When using multiple parallel equipment-state machines the state of an equipment/control module will propagate up to the equipment state machine of the unit, e.g. if a level sensor breaks the state of the unit should go to an error state to indicate there is something wrong within the unit.

The normal execution of an operation causes the equipment-state ma-

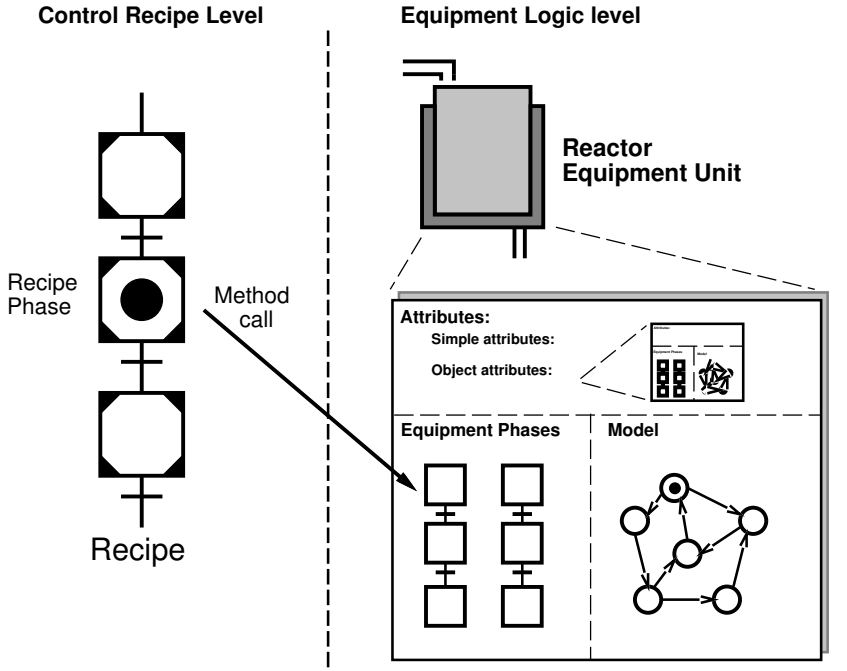

Figure 3.6 Equipment unit with finite state machine modeling its state.

chine to change state, see Fig. 3.9. For example, when the control system sends a signal to a valve to open, the equipment-state machine of the valve will go from the state `Closed` to the state `Open`.

The equipment-state machine serves two purposes. The first purpose is to be able to check that all the equipment objects are in a consistent state when a method is invoked. For example, it should not be allowed to open a valve if the valve already is open, and it should not be allowed to fill an equipment vessel that is already full. In a properly designed batch control system, which always executes in automatic mode, one could argue that consistency checking of this type is already performed through off-line validation and verification of recipes, equipment logic, and production schedules. However, in practice batch processes are often run in manual mode for substantial parts of time. Then, it is the operator that manually invokes different equipment phases and a consistency check of the proposed type could be very useful. The consistency check is realized by associating a set of allowed states (or one state if a single state machine is used) with each operation in the equipment control. It is only allowed to start the execution of an operation if the state of the equipment unit

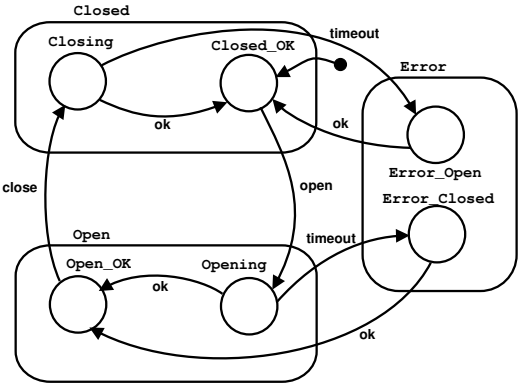

Figure 3.7 The equipment-state machine of a valve.

belongs to the allowed set of states. The consistency check is implemented using a *start-state machine* associated with each operation, see Fig. 3.10. In the figure the start-state machine, named `Start`, is in the `Not_OK` state, which means that the operation that the start-state machine belongs to is not allowed to start if it is called at this time. For example, if the operation is a heating operation, the reason that it is not allowed to start might be that the temperature of the unit is too high.

The second purpose of the equipment-state machine is to provide a structure for organizing the safety and supervision logic at the equipment control level. This is done by implementing the safety logic as transitions or guards in the equipment-state machine, as in Fig. 3.11. The safety logic expressed in a transition is only enabled when its preceding state is active. If a fault occurs, the safety logic causes a state transition from a normal state to a fault state. For example, when the valve in Fig. 3.7 receives a signal from the control system to open, the state changes to

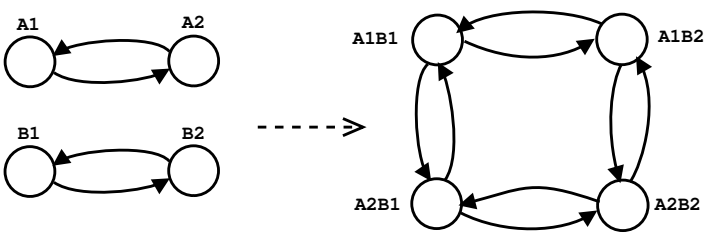

Figure 3.8 Two automata composed to a single automaton.

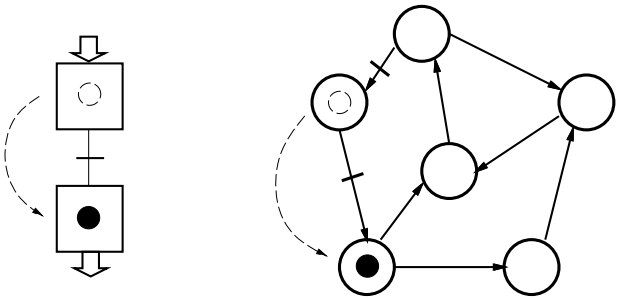

Figure 3.9 The execution of an operation, implemented with a Grafchart procedure, changes the state of an equipment-state machine.

`Opening`. The error transitions of this state will become active. One of these error transition conditions is that the valve does not respond to the control signal within a given time. If a signal is not sent back that the valve is physically open within this time, the equipment-state machine of the valve will go to the `Error_Closed` state, representing that the valve is stuck in the closed position. There might be a large difference if the valve fails to respond to a command when it is in the `Open` state compared to if the same problem occurs when it is in the `Closed` state. Which is the most severe error depends on the process, e.g., if the valve is for the cooling water of a jacketed exothermic reactor a failure in the `Closed` state is probably worse than a failure in the `Open` state and in a different process the opposite would be true instead.

The state machines can be implemented in several ways. In the implementation in this thesis multiple input multiple output (MIMO) macro steps have been used, see Fig. 3.12. A MIMO macro step have several input and output ports, similar to the *superstate* in Statecharts [Harel, 1987]. Using the MIMO functionality of the macro step it is possible and convenient to model hierarchical state machines of the proposed type in JGrafchart.

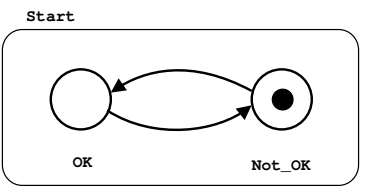

Figure 3.10 The start-state machine of an operation for the consistency check for the start of the operation.

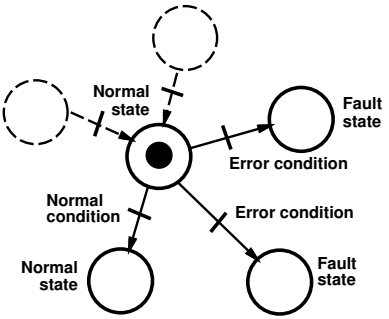

Figure 3.11 State machine with safety and supervision logic.

A hierarchical state machine is convenient to use when modeling the errors of an equipment object. For example, when producing a product in a reactor it might be important not to exceed a given temperature to maintain quality. The reactor tank also has hard constraints on what its

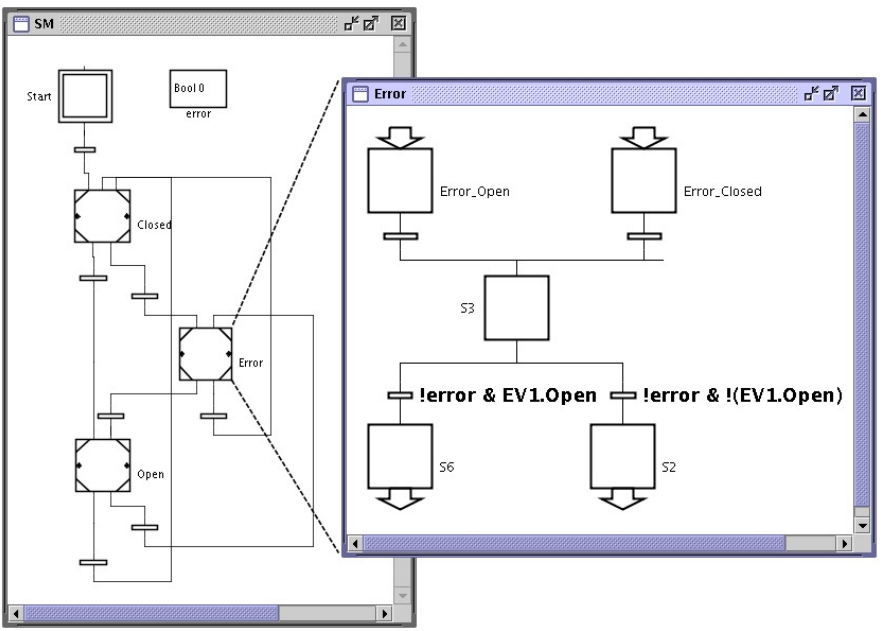

Figure 3.12 Equipment-state machine of a valve implemented in JGrafchart using MIMO macro steps.

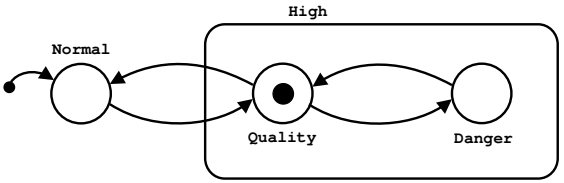

Figure 3.13 Equipment-state machine of a temperature sensor

operating range is, e.g., maximum pressure and temperature. This results in different levels of severity of exceptions. A bad product is not nearly as important as the risk of causing human injury. The different error states would typically result in different alarms to the operator. If there is a risk that the quality will go out of the specifications there will be a warning to the operator, but if the unit is going to a safety-critical state the unit needs to immediately be taken to a safe state by the exception handling. The equipment-state machine of a temperature sensor would typically look as in Fig. 3.13. The `High` state of the state machine contains the two states `Quality` and `Danger`, which corresponds to the states described above. The same equipment-state machine for the temperature sensor implemented in JGrafchart is shown in Fig. 3.14. The conditions in the transitions of the state machines are dependent on the recipe. How the conditions can be set is described in Section 3.4. The equipment-state machine of a temperature sensor may also have many other error states, but they are not part of this example.

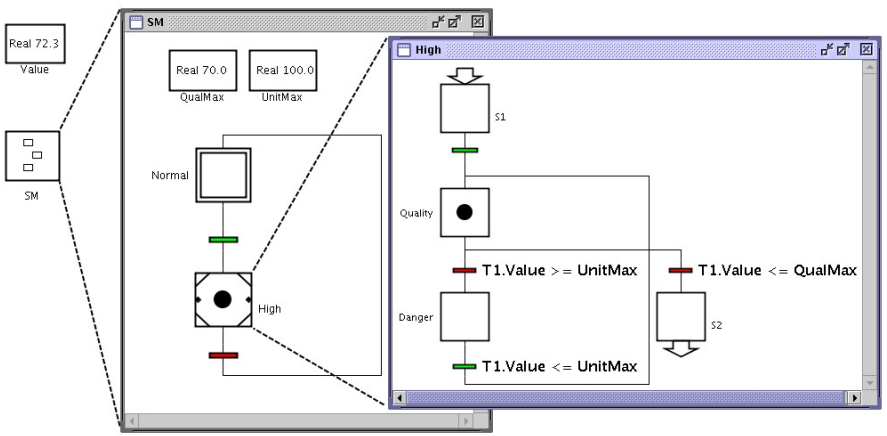

Figure 3.14 Equipment-state machine of a temperature sensor (`T1`) implemented in JGrafchart using a macro step.

Exception Handling Structure

In the proposed structure for exception handling most of the functionality is associated with the equipment operations and phases. Each equipment operation, e.g., `Charge`, `Heat`, and `Clean`, is implemented in JGrafchart using a workspace object on the workspace of a unit, see Fig 3.15. Also on the unit workspace are the equipment/control modules, the equipment-state machine of the unit, and attributes describing if the unit is available or if any error has occurred in the unit.

Each operation workspace object contains a procedure (i.e. the sequential control), the *procedure-state machine* describing the state of the procedure according to Fig. 2.4, the start-state machine (for the consistency check), and an *exception handling workspace*, see Fig 3.16.

The procedure of an equipment operation holds not only the equipment sequential control, but also contains several checks, which need to be performed when a procedure is called from a recipe. First it checks if the unit is available (in S88 only one operation at a time is allowed to be

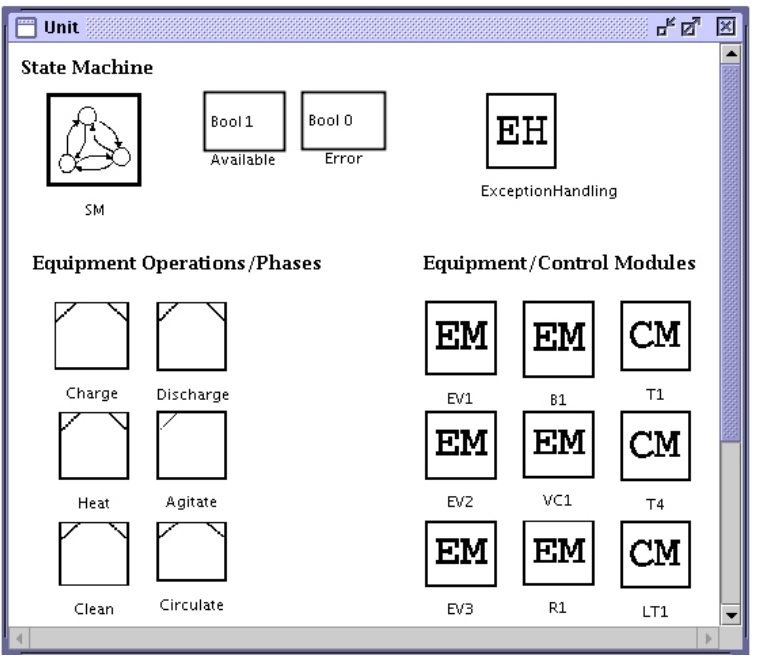

Figure 3.15 The workspace of a unit with equipment-state machine, equipment operations/phases, equipment/control modules, and unit exception handling. Attributes describing if the unit is available or if any error has occurred in the unit are also shown.

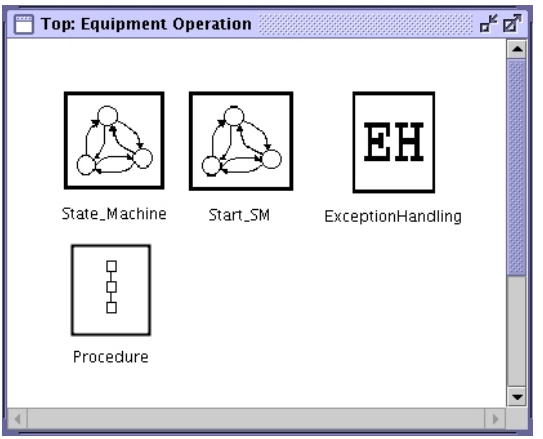

Figure 3.16 Equipment operation workspace.

active in a unit) by checking the state of the equipment-state machine of the unit or the attribute `Available`. It reserves the unit if it is available and then checks if the procedure-state machine of the procedure itself is in the `Idle` state and if so changes the state to `Running`. The check if the unit is in a consistent state at the start of the operation is also performed here using the start-state machine. This could be implemented in several ways, one example is to implement it in JGrafchart according to Fig 3.17. In the figure the operation called is `Charge`, the signal to reserve the unit is `reserve`, `SM` is the name of the equipment-state machine describing the state of the unit, `LT1` is a level sensor, `EV1` is a valve, and the check if the unit is in a consistent state when starting the operation is a start-state machine named `Start`. The implementation of the start-state machine in JGrafchart is shown in Fig 3.18. The start-state machine consists of only two steps, `OK` and `Not_OK`. The two states are mutually exclusive and if the `OK` state is active the operation is allowed to start. In this small example the unit is not allowed to be full and the in-valve (`EV1`) is not allowed to be open for the operation `Charge` to start.

The procedure-state machine of the operation can be implemented in the same way as the state machine of an equipment object using MIMO macro steps. The procedure-state machine can be used by an operator to, e.g., pause the execution of the operation or just as a display object to depict in which state an operation is. The following twelve procedural states from S88 described in Fig 2.4 are modeled in Fig 3.19: `idle`, `running`, `complete`, `pausing`, `paused`, `holding`, `held`, `restarting`, `stopping`, `stopped`, `aborting`, and `aborted`.

In this implementation a large part of the exception handling is specific

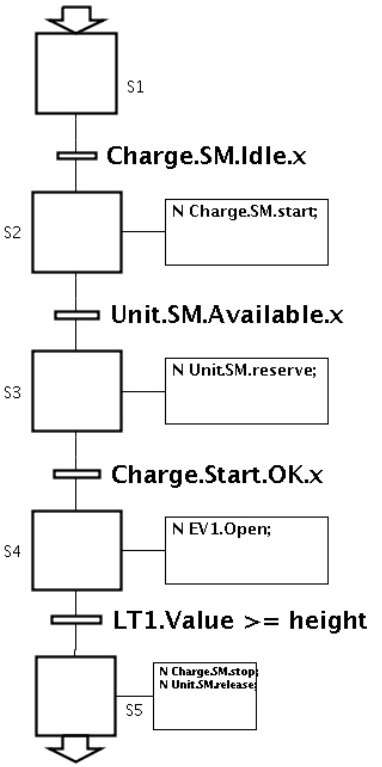

Figure 3.17 Examples of checks and control in a charge operation implemented in JGrafchart. The operation checks if the operation is in the `Idle` state, changes the state of the operation to `Running` by sending the signal `start`. It reserves the unit if it is available, checks if the start of the operation is allowed by the state of the unit and then opens the in-valve `EV1`. When the level is above the specified height the valve is closed, the state of the procedure-state machine is set to `Idle`, and the unit is released.

to an operation. The exception handling workspace of the operation holds both the recipe level, and the equipment level exception handling logic, see Fig 3.20. Both of these would be running in the Safety Instrumented System in a control system implementation for a real plant. The recipe level exception handling logic will be further discussed in Section 3.4.

In the equipment level exception handling logic, two types of operations can be identified:

- Detection logic, based on the equipment-state machines of the unit and the equipment/control modules.

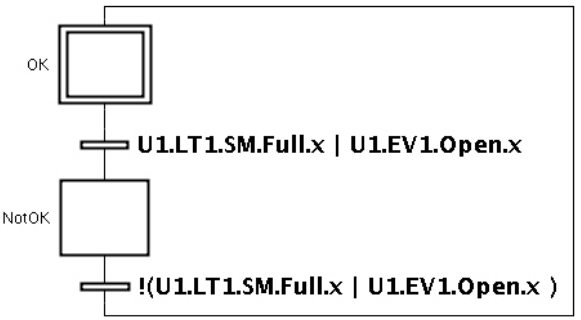

Figure 3.18 The start-state machine of the `Charge` operation. The two states important to the start of this operation is that the unit is not full and that the in-valve (`EV1`) is not already open.

- Logic to handle the exceptions. Implemented much like a recipe where appropriate actions are taken depending on the specific exception detected.

Both kinds of operations will be called from the recipe level exception handling logic. The detection logic, `Detection` in this example, checks the state of the unit by looking at either only the unit's equipment-state machine, or at the equipment objects' individual equipment-state machines depending on how the states of the equipment objects propagate to the unit. One way to implement the propagation of exceptions is to let the unit's equipment-state machine only consist of the states `OK` and `Error`, and let the detection logic be trigged by the unit's equipment-state machine, see Fig. 3.21.

The operations to handle the exceptions in Fig. 3.20 are:

- `StartExc` - the sequence, which takes the unit back to a state where the operation is allowed to start.
- `Emergency` - emergency shut-down (could be the same for all operations in a unit).
- `Exc1-Exc2` - handling of exceptions 1 and 2, which can be any exceptions specified.

Not all of the exception handling is associated with the execution of operations. The unit exception handling is running all the time, see Fig. 3.22. There is, e.g., exception detection logic checking the state of the equipment modules even though there is not any operation running in the unit. The unit-specific exception handling and the operation-specific exception handling logic need to be synchronized to avoid false alarms.

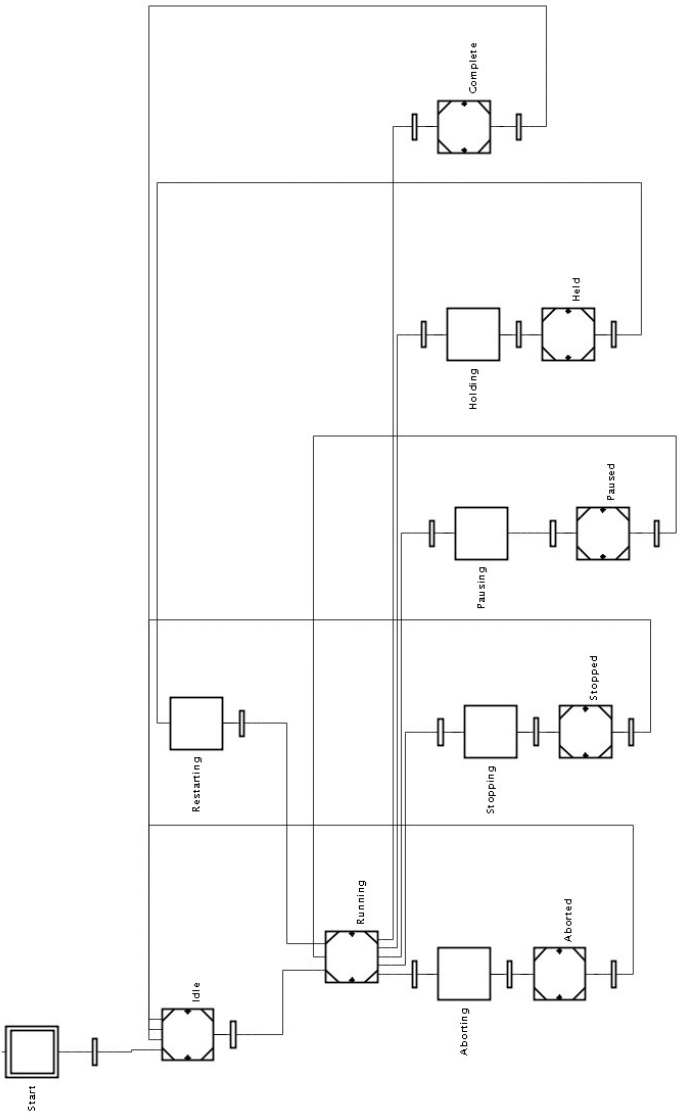

Figure 3.19 The states of an equipment operation modeled with MIMO macro steps.

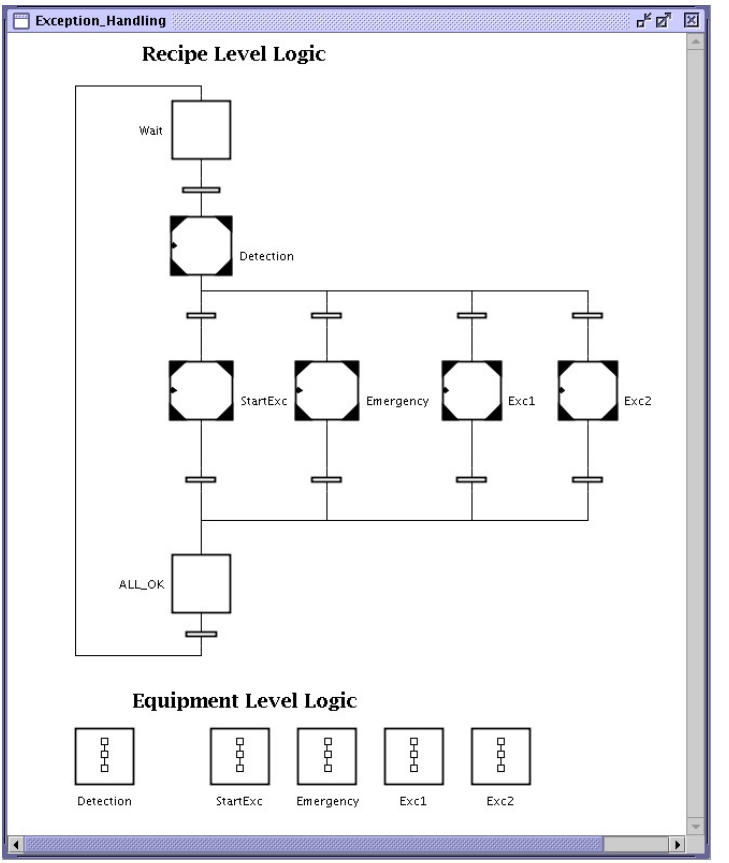

Figure 3.20 Exception handling associated with an operation.

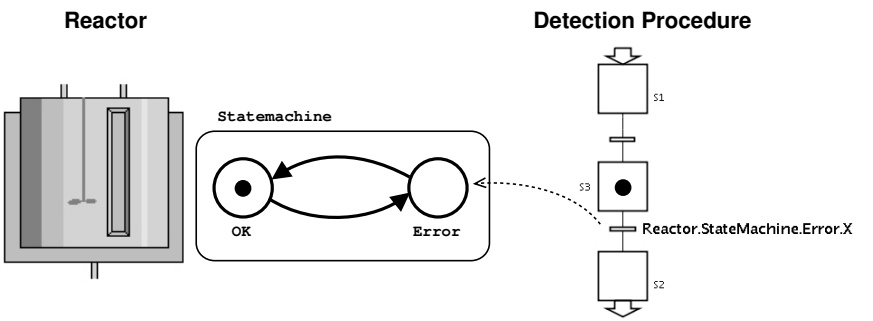

Figure 3.21 Implementation of detection logic for an operation in a reactor unit.

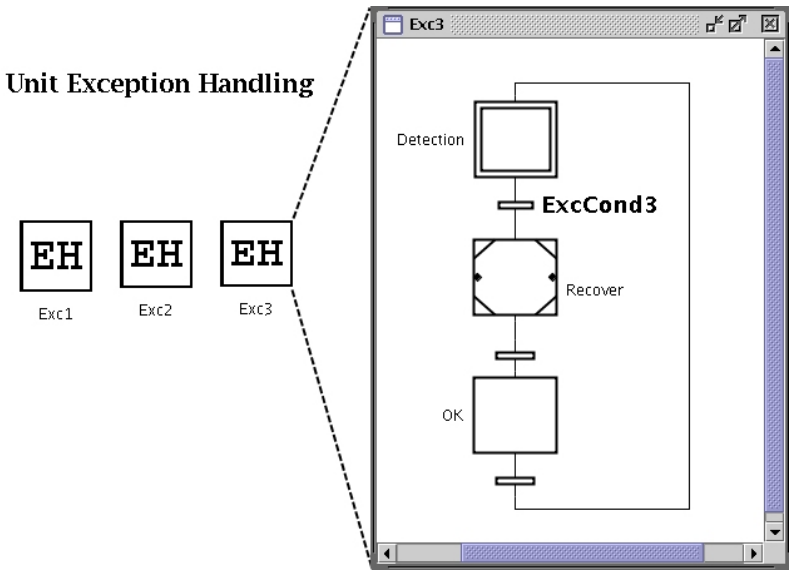

Figure 3.22 Unit detection and exception handling logic. Three different workspace object for the exceptions `Exc1-Exc3` are shown. `ExcCond3` is the condition that triggers the exception handling for `Exc3`. The detection is based on the equipment-state machines for the equipment/control modules in the unit.

3.4 Recipe Level Exception Handling

In the proposed approach the main responsibility for fault detection and exception handling lies at the equipment control level. However, exception handling is also needed at the recipe level. For example, an exception that has occurred must be fed back to the control recipe, recorded in the batch report, and appropriate actions must be taken. If a batch is aborted the scheduler needs to reschedule the batch for a later time.

An important consideration is how to separate the recipe information from the exception handling logic and operations. If the latter is included in the recipe, it becomes difficult to develop, maintain, and use. The exception handling would probably be several times larger than the recipe itself. There is only one correct way to produce a batch, but the process may fail in almost an infinite number of ways. Grafchart provides several features that simplify the representation of exception handling logic at the recipe level.

It is possible to use exception transitions for recipe level exception handling. An exception transition is a special type of transition that may

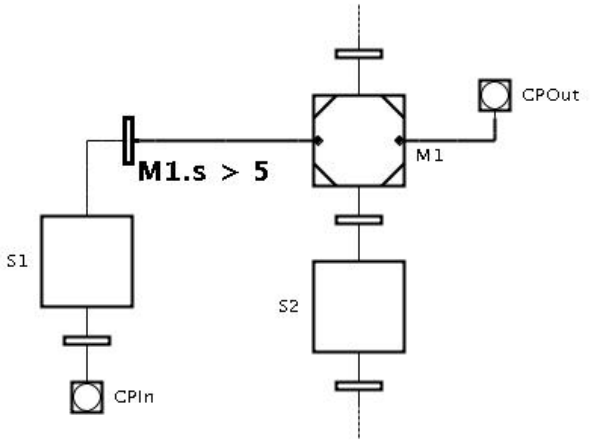

Figure 3.23 An exception transition connected to the macro step M1.

be connected to a macro step or a procedure step. The exception transition is connected to the left hand side of the macro (procedure) step. An ordinary transition connected to a macro (procedure) step does not become enabled until the execution of the macro (procedure) step has reached the exit step. An exception transition, however, is enabled all the time while the macro (procedure) step is active. When the transition is fired the execution inside the macro (procedure) step is aborted and the step succeeding the exception transition becomes activated. Exception transitions have priority over ordinary transitions in cases where both transitions are fireable at the same time. An exception transition connected to a macro step is shown in Fig. 3.23. The exception transition will fire when M1 has been active longer than 5 seconds.

In the recipe-level exception handling the exception transitions are connected to the procedure steps representing the control recipe operations, see Fig. 3.24. At least two extra graphical objects, an exception transition and a connection post, are needed for each procedure step in the control recipe. This makes the control recipe become very much larger than compared to when it is only holding the logic for normal operation. The detection logic of the recipe level exception handling is in the control recipe itself while the handling of the exception can be stored somewhere else.

Another possibility, to avoid the extra graphical objects in the recipe, is to use step fusion sets [Jensen and Rozenberg, 1991]. Step fusion sets make it possible to have different graphical representations of the same

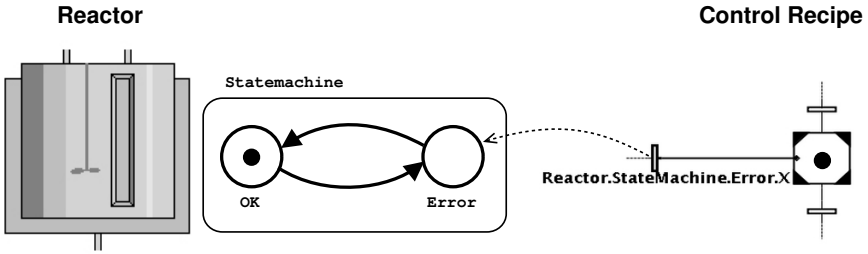

Figure 3.24 Exception transitions for recipe level exception handling.

step. The steps in a step fusion set can be seen as different views of the same step, which are separated and put at different locations as shown in Fig. 3.25. This way sequences can be divided into smaller, easier to read, parts. The steps in a step fusion set do not have to be of the same kind, e.g, a macro step and a procedure can be in the same step fusion set. When one of the steps in the set become active, all the steps in the set become active.

A step fusion set can be either abortive or non-abortive. If the step fusion set is abortive an exit transition of a step becomes enabled when the step reaches its exit step. If the transition condition becomes true the transition fires and all the steps in the step fusion set becomes inactive. Procedure and macro steps will abort their execution and the steps' abortive actions will be executed. The abortive actions have to be taken in to account when designing macro steps and procedures. If the step fusion set is non-abortive all the steps in the step fusion set have to reach their exit step before the exit transitions become enabled.

The step fusion set approach is based on letting the procedure step that calls an operation in the control recipe be in the same step fusion set as the procedure step that calls the detection logic of the operation, i.e., the procedure step named `Detection` in Fig 3.20. Consider a control recipe consisting of a sequence of procedure steps (`I`) making procedure calls to different equipment operations (`II`), see Fig. 3.26. The transition after the procedure step in the control recipe (`III`) becomes enabled when the execution of the corresponding operation is finished. The procedure step in the control recipe has a corresponding procedure step in the recipe level exception handling logic (`IV`), these two steps are in the same step fusion set. The procedure step in the recipe level exception handling logic calls the detection operation at the equipment level (`V`). If an exception occurs before the operation in the control recipe is finished, the equipment level exception handling logic detects it. The detection of an exception enables

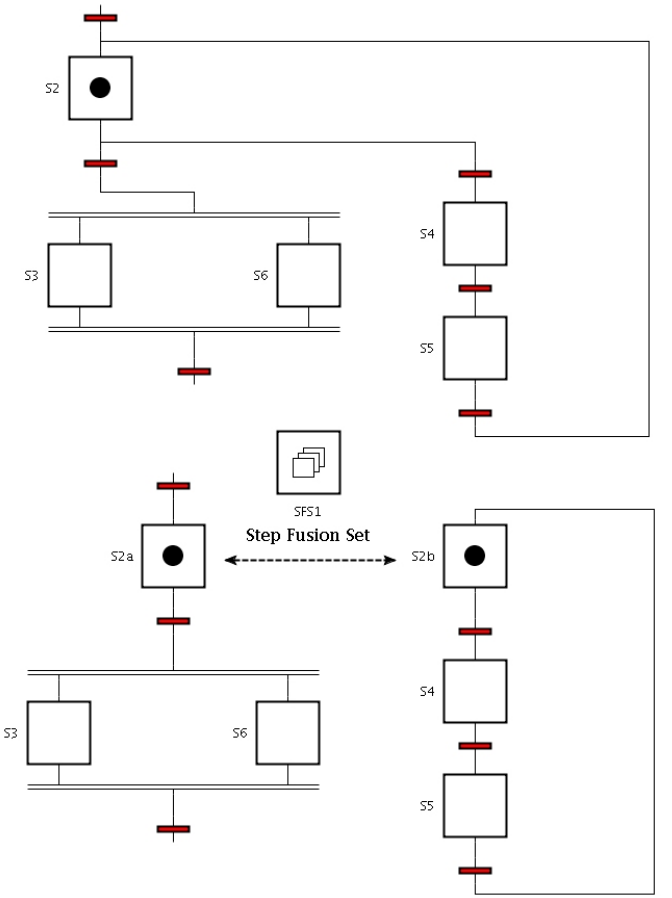

Figure 3.25 Step S2 separated into S2a andS2b using a step fusion set. If the transition after S2b fires S2a and S2b becomes inactive and S4 becomes active. If the transitions after S4 and S4 become true, S2a and S2b become active again.

the transitions connected to the out-port of the detection procedure step (VI Error Exits). The transition associated with the specific exception that has occurred becomes true, and the operation for the handling of the specific exception starts. If the step fusion set is abortive the execution of the operation called from the control recipe is aborted and the abortive actions of the procedure step in the control recipe are executed.

The first two error exits would typically be the exit for emergency shutdown of the unit (VII) and the exit for when the starting state is not a member of the allowed starting states (VIII). Other error exits would be

for the malfunction of a valve, a sensor, or any other equipment belonging to the unit. A default operation, which takes the unit to a fail-safe state if an unspecified exception occurs should also be implemented. The operations would generate alarms for the operator and other information during the execution of the exception handling logic. The operations for the exception handling may be automatic, semi-automatic, or manual.

The nature of the actions that must be taken depends on the application. In a few very special cases it might be possible to "undo" an operation and rollback the execution of the recipe to a safe execution point, and from there continue the execution using, e.g., a new unit. This is similar to the check-pointing and rollback employed in fault-tolerant real-time systems [Jalote, 1994]. One situation where it would be natural to be able to rollback the execution is when an operation is called and the equipment object is not in a allowed state for the operation to start. When the state of the equipment object is changed to one of the allowed states, the operation would be restarted and the execution of the recipe would be able to continue.

If Fig. 3.26 is used as an example, the steps of the exception handling would be:

- The detection logic in `Detection (V)` is triggered by that the state of the unit is inconsistent with the start of the operation.
- The condition of the error exit for the start exception would become true and the `StartExc (VIII)` operation is called from the recipe level exception handling.
- The `StartExc (VIII)` operation takes the unit to an allowed state for the start of the original operation, called from the control recipe, and the step `ALL_OK (IX)` will become active.
- The exception recipe would make a `Rollback (X)` and restart the detection operation and the operation in the control recipe, since the two steps are in the same step fusion set.

However, due to the nature of chemical batch processes a rollback is in most cases not a viable alternative. For example, it is very seldom possible to undo a chemical reaction. Also in the more common case where the batch cannot be produced as intended there are several alternatives. In certain situations it might be possible to still make use of the batch to produce a product of a different grade or quality. In other situations it is possible to recirculate the batch ingredients for later reuse. Also in the case where the batch cannot be used as a product, special actions must be taken. Due to environmental regulations the partly produced batch

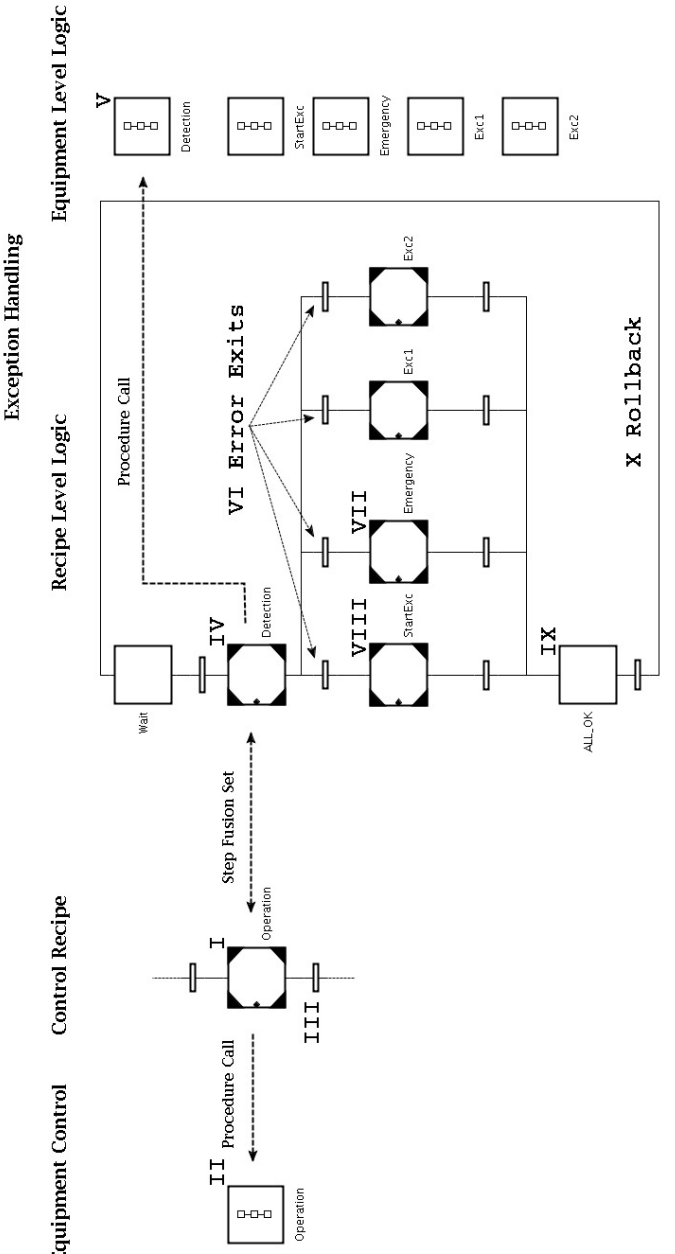

Figure 3.26 Step fusion sets for exception handling: control recipe and exception handling.

must be taken care of in an appropriate way. This may include further processing to separate or destroy the batch ingredients.

One problem, which occurs when an operation is restarted after an exception is taken care of, and the normal execution should continue, is that in the implementation described above the operation will start from the beginning. It will try to reserve the unit again, *etc*, as described in Fig. 3.17. Since the unit is already reserved by the operation itself the check needs to be overridden and some manual control by the operator is necessary. One way to take care of this problem is to change the linking between the recipe and the equipment logic to a lower level. The recipe operation is divided into several recipe phases or even smaller parts (e.g. the reservation of the unit), see Fig. 3.27. Each of the recipe phases makes a procedure call to the corresponding phase at the equipment control level.

In this approach, since the number of phases are increased the number of workspaces containing exception handling is increased. One workspace for each phase is needed, each only containing the exception handling needed for the specific phase. For example, it is not necessary to have exception handling logic that deals with the reservation of a unit once it is reserved. The detection procedure becomes more specific and the number of errors to be handled in each exception handling workspace is decreased. If the operations are divided into smaller parts some of the phases can be reused by other operations. For example, the procedures for reserving and releasing a unit are unit specific and could be reused by all the operations belonging to the same unit.

In the implementation described above the separation between the recipe level and the equipment control level exception is at the operation level using procedure steps and procedures in JGrafchart. The separation of the exception handling is at the same level as the separation in the procedural control and follows the S88 standard. The separation could be at any level specified in S88, see Fig. 2.3.

Another way to implement the exception handling is to have the exception detection logic in the equipment operations, see Fig. 3.28. If an exception occurs it will be detected by the exception conditions in the operation. The exception conditions are specific to which step is active in the operation and are based on the equipment-state machines of the unit and the equipment/control modules. The normal execution of the operation is aborted and the operation will finish using a different path. Since an exception has occurred the transition after the procedure in the control recipe will not be fireable. Instead the error exit corresponding to the exception is fired and the exception handling will try to recover from the exception, see Fig. 3.29, where this is implemented using non-abortive step fusion sets. In this approach the recipe level and equipment control level exception handling are no longer separated in the same way as

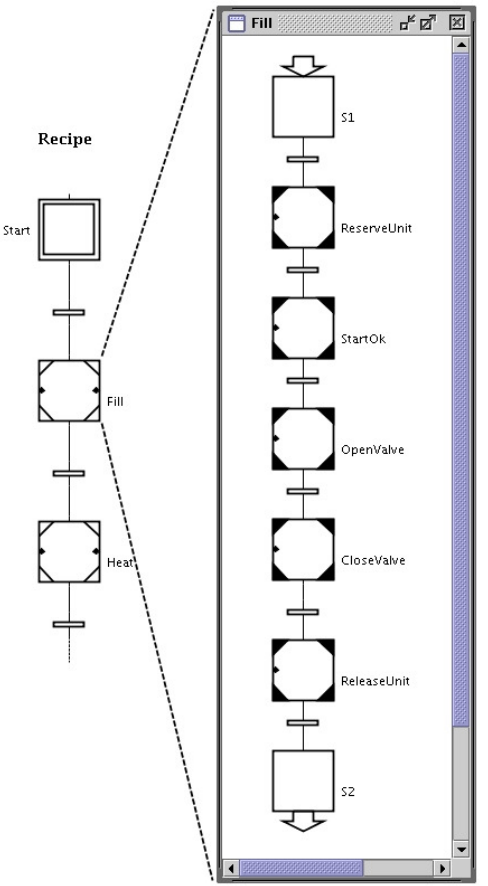

Figure 3.27 A recipe operation divided into phases and smaller parts. Each of the procedure steps calls a procedure at the equipment control level.

the normal procedural control, i.e, the control recipe and the equipment control.

Recipe Dependent Conditions

Units are usually used for the manufacturing of several different products and different grades of the products. The recipe parameters, e.g., size, temperature of reaction, duration of mixing, and catalyst, are therefore changed according to the specifications of the product. However, it is equally important to change the parameters of the exception handling, i.e., the conditions in the equipment state machines of the different equip-

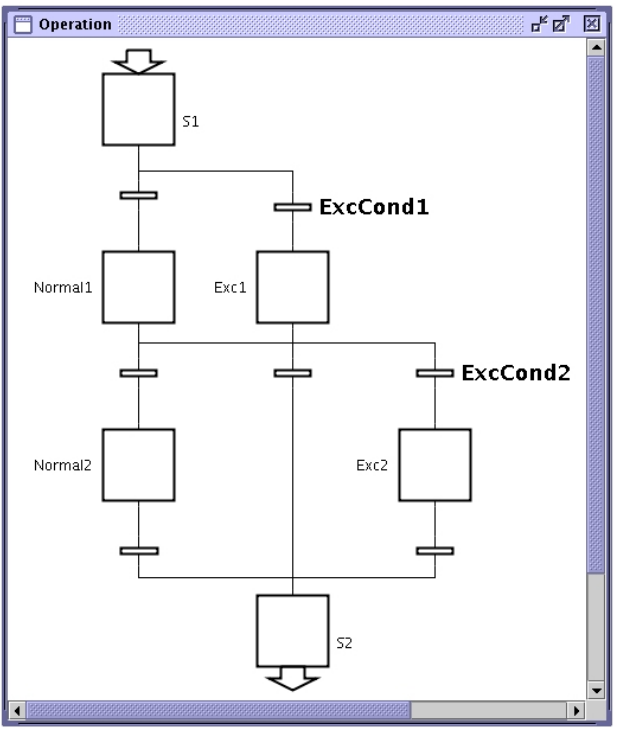

Figure 3.28 Exception detection logic in an equipment operation. If an exception occurs the execution of the operation will finish but by following the normal path. The condition of transition after the procedure step in the control recipe will not be true. Instead the error exit in Fig. 3.29 corresponding to the error, which has occurred is fired.

ment/control modules. For example, what was considered a normal temperature when producing one product might lead to a run-away reaction when producing another product. The conditions of the transitions can easily be changed by using stored actions in the procedure step, which calls an operation, to change the variables used in the conditions of the transitions in the state machine. In the same way exit actions can set the variables to, e.g., the default values of the unit, when a procedure is finished. When the procedure step in Fig. 3.30 is activated the variable `QualMax` in the state machine will get the value `90.0`. This means that if the temperature is not changed the state will go to `Normal`, since the value of the temperature sensor is less than the value of `QualMax`.

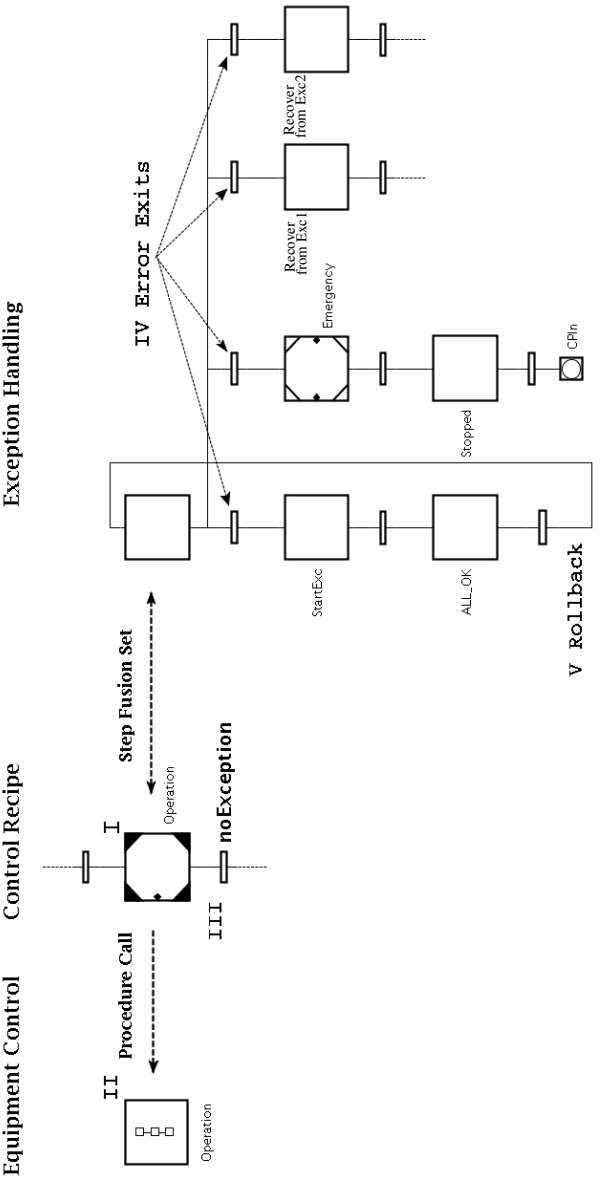

Figure 3.29 A procedure step (I) calls the operation (II) with the detection logic, see Fig. 3.28. The transition (III) after the procedure step will only fire if no exception is detected. If an exception is detected the corresponding error exit (IV) will fire and the exception handling will try to recover from the exception.

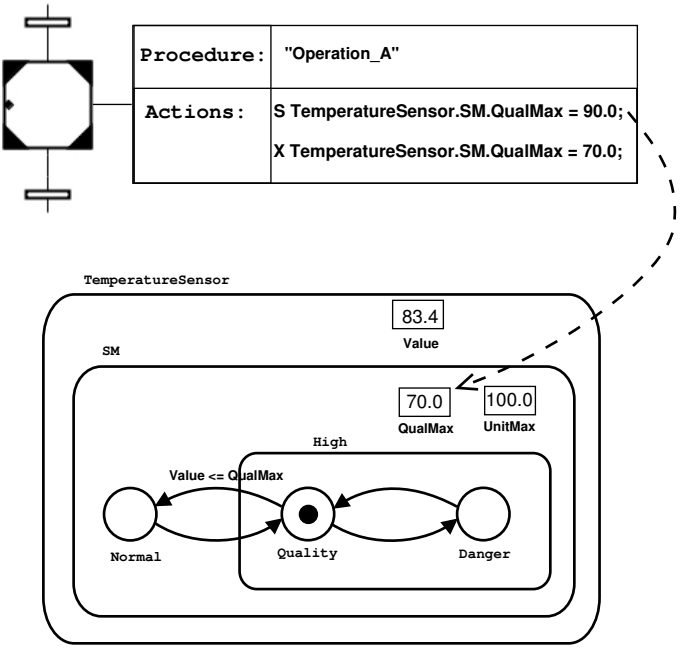

Figure 3.30 Conditions in state machine set by a procedure step.

3.5 Synchronization

One important part that is often hard to implement in batch control systems is the synchronization between units. In the recipe, two operations seem to be simultaneous, but at the equipment control level synchronization is needed. For example, to make sure that valves and pumps are opened/closed and started/stopped in the correct order during a transfer from one unit to another, the units need to perform a handshake. If the pump used for the transfer is a displacement pump and it is started before the valve is opened, there is a risk that the pipe will burst. The handshake works as an interlock to make sure the actions are taken in the right order. In Fig 3.31 an implementation of a handshake between a transfer operation and a receive operation in two different units is shown.

The task of synchronization becomes even harder when also the recipe level exception handling should be synchronized with the normal parallel recipe operations, i.e. concurrent operations in the recipe. In Fig 3.32 the

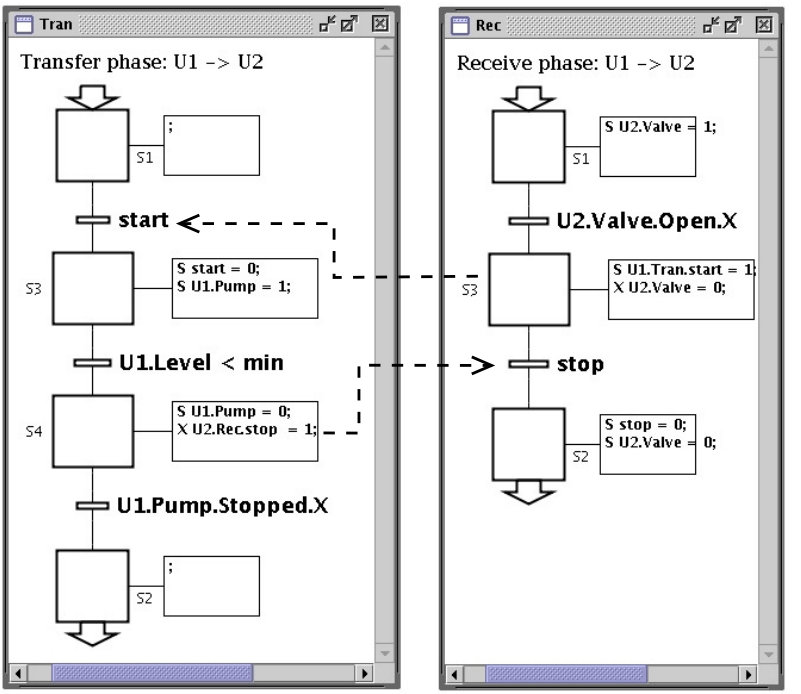

Figure 3.31 Synchronization using handshake between operations in two units.

transfer operation of `Unit1` and the receive operation of `Unit2` are shown at the recipe level as well as the procedure step representing the detection operations in the recipe level exception handling in both units. The four procedure steps in the figure will be activated at the same time instance. If an exception occurs in `Unit1` during the execution, the exception handling for `Unit1` will be activated. If `Unit2` does not receive this information it will continue to try to transfer material. `Unit2` needs to know which kind of fault has occurred in `Unit1`. The two units need to perform a controlled abortion of the two operations in a handshake manner just like when starting and stopping the normal execution of the two operations. The result is that the exception logic for the errors in the receive operation must mirror the exception handling logic for the corresponding transfer operation (and the other way around) to perform a controlled abortion of the two operations. Exception handling logic for exceptions occurring in `Unit1` must be part of the exception handling logic of `Unit2` and vice versa.

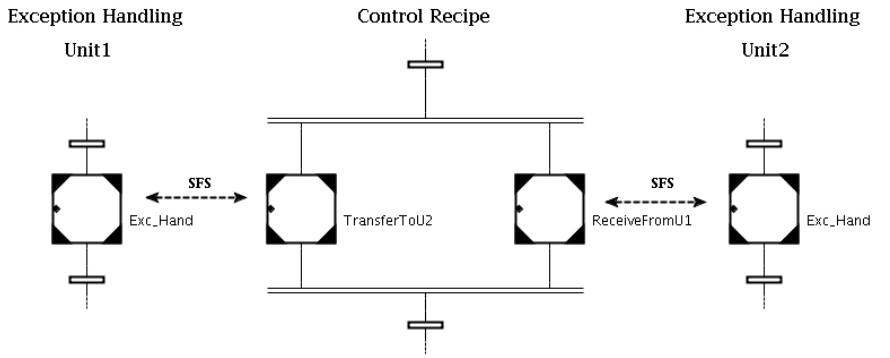

Figure 3.32 Transfer and receive operations in a recipe with exception handling.

3.6 Summary

A new approach to structure exception handling in recipe-based batch control has been described in this chapter. The structure is based on augmenting units, equipment, and procedural elements with a state-machine based model. The state machine models the current state of the object and also organizes which new states the object can reach from the current state. The models describe both normal operation states and faulty states. This makes it easy to model that certain faults can only occur from specific states and at specific times during operation. A new language object, the MIMO macro step, has been added to Grafchart to be able to easily implement hierarchical state machines. The MIMO macro step can have several inputs and outputs and encapsulate a sub-level in the state machines.

State machines have been used to model different kinds of behavior in the batch control system. Start-state machines are used to decide if the unit is in a state where an operation is allowed to start. The decision is based on both measurements of and the availability of auxiliary equipment needed for the operation. The consistency checks are realized by associating an allowed set of states with each operation in the control system. Procedure-state machines model the state of an operation, i.e., if the operation, e.g., is running, stopped, or idle. Equipment-state machines are modeling the state of each equipment module, such as a valve, a pump, or the whole unit.

During the manufacturing of a batch the development of the recipe is dependent on the state of the unit. If an operation is not allowed to start or if it has to be aborted the recipe needs to act accordingly to

inform the operators and ensure safety. This logic taking care of problems arising during execution is normally hidden from the operators during normal operation and separated from the normal recipe. Information to higher levels such as production planning and scheduling is also needed. This is described in the next chapter. Several ways of separating normal recipe execution from exception handling logic at the recipe level have been discussed in this chapter. One way is to use exception transitions, a special transition that can fire at any time and abort the execution of any kind of step. A new language object, step fusion sets, is added to Grafchart to allow for a new way of making this separation easier. The step fusion set holds several steps representing different views of the same step. If one of the steps are activated all the steps in the step fusion set becomes active. The step fusion set can be either abortive or non-abortive. A recipe can easily set the conditions in the state machines to be enable logic to detect recipe dependent exceptions and to change normal operating regions and limits.

The described structures and language elements in JGrafchart should make the development, maintenance, and use of the exception handling logic both at the unit level and at the recipe level an easier task. The new structures and concepts described in this chapter also fits nicely into the S88 batch control standard.

4

Exception Handling Applied to the Procel Batch Pilot Plant

4.1 Introduction

In this chapter the implementation of a batch control system for the Procel batch pilot plant is described. The EC/GROWTH project CHEM has developed toolboxes for batch control, which are used in the control system. A closed-loop framework is presented that integrates decision support tools required at the different levels of a hierarchical batch control system [Musulin *et al.*, 2005]. The proposed framework consists of a reactive batch scheduler (MOPP) and a fault diagnosis system (ExSit-M) developed by the Department of Chemical Engineering at Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, and a S88-recipe-based coordinator implemented in JGrafchart, see Chapter 3.2. The tools need to exchange information to obtain optimal utilization of the production plant. The complete integrated system is built using the S88 batch control standard [ANSI/ISA, 1995].

The integration of a fault diagnosis system (FDS) aims to timely provide the process state information to the different levels in the decision-making hierarchical structure, thus reducing the risk of accidents and improving the efficiency of the reactive scheduling in the most effective way. To handle unit supervision, exception handling, and recipe execution a coordinator is implemented in JGrafchart. The unit supervision is based on modeling the state of each equipment object and procedural element using finite state machines as described in Chapter 3. A closed-

loop framework for on-line scheduling of batch chemical plants integrating both robustness considerations, fault diagnosis, recipe coordination, and exception handling is proposed in this work. This on-line integration leads to a fast execution of the recovery procedures and the rescheduling. The work in this chapter is a continuation of the case study presented in [Olsson, 2002].

4.2 CHEM - Advanced Decision Support System for Chemical/Petrochemical Manufacturing Processes

The aim of the CHEM project was to develop and implement an advanced Decision Support System (DSS) for process monitoring, data and event analysis, and operation support in industrial processes. The system is an integration of software tools, which improve safety, product quality, and operation reliability as well as reduce economic loss due to faulty states in refining, chemical, and petrochemical processes.

Advanced methods and software tools based on statistical, system theoretic, and artificial intelligence methods for process monitoring, detection, diagnosis, and decision have been developed within the CHEM project. These tools have been integrated into the DSS in a modular fashion. The DSS is developed to be able to interface with commercial plant database and process control software. The DSS environment was developed and tested at pilot plants and industrial sites.

The main motivations for the CHEM project were as follows:

- The great amount of information collected at industrial plants should be used to improve efficiency and productivity, to avoid unscheduled shutdowns and abnormal situations. Considering the economic loss and the potential damages, a system with the capability to handle those situations will have big impact for the economics as well as for safety and environment protection.

- The complexity of control systems makes it more and more difficult to make the right decisions at the right time. By giving easy to use tools to the operators they will more easily be able to make the right decisions during operation of the plant. The tools should provide explanations, thus increasing the level of knowledge of the process.

- A lot of work has been conducted in similar fields: process trend analysis, fault diagnosis and decision support systems, but many methods still work separately and it remains difficult for developers

of supervision systems to build applications for new plants. Therefore, it is necessary to join the efforts of specialists with a wide spectrum of expertise area to be able to build a comprehensive supervision system.

The DSS demonstrator is built around G2 [Gensym Corporation, 1995], a commercial software, from Gensym, widely used in chemical and petrochemical industry. G2 is used as the integration tool even though some toolboxes are developed in other languages.

4.3 Toolboxes

In this section the different toolboxes that have been integrated for the Procel control system are described. In Fig. 4.1 an overview of the different toolboxes and the flow of information can be found.

CHEM Integration Platform

The CHEM integration platform is developed in G2. The platform consists of two layers, CHEM Communication Manager (CCOM) and a Data Manager (DTM) described below, see Fig. 4.1.

CHEM Communication Manager CCOM allows other toolboxes to communicate through the exchange of messages. It is based on a public

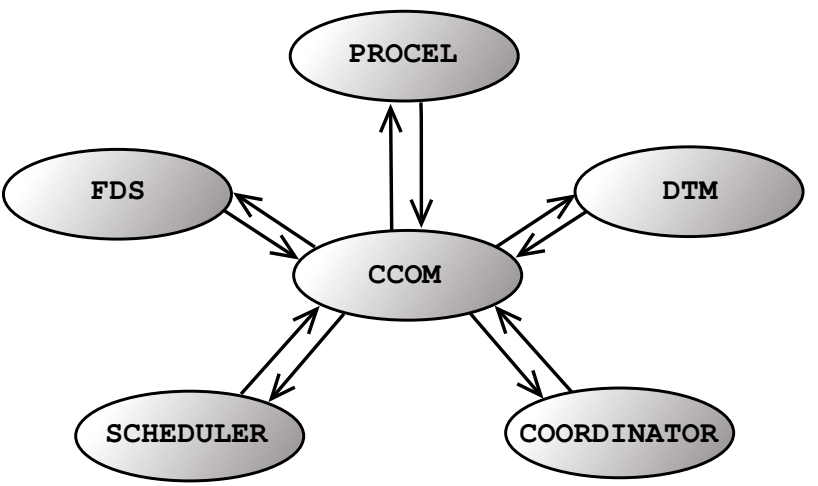

Figure 4.1 Overview of and the information flow between the different toolboxes in the Procel control system.

domain Message Oriented Middleware (MOM) software, xmlBlaster [xmlBlaster.org, 1999], that provides Publish/Subscribe and Point-to-Point message communication. CCOM acts as a server that clients, e.g., a toolbox, can connect to. A client API has been developed on top of the MOM interface to provide additional functionality and hide the aspects of transport protocols to the clients. The messages are in the XML format.

Data Manager To make the exchange of data between the components/toolboxes easier a Data Manager (DTM) has been developed. The DTM is used to store and retrieve data produced or requiered by the different toolboxes. It is a self-contained component that is seen by the other components as an application offering a number of services, which are accessed through CCOM, see Fig. 4.1.

Scheduler

MOPP is a scheduler package developed at UPC [Cantón Padilla, 2003; Graells *et al.*, 1998; Ruiz *et al.*, 2001; Arbiza *et al.*, 2003a; Bonfill *et al.*, 2004]. The scheduler uses Event Operation Networks (EON) to model the system. EON models have proved to be an efficient way to describe time constraints between operations in complex production structures. The EON model is built using a general recipe description and other guidelines from the batch control standard S88.

MOPP has a library of different dispatching rules to determine a feasible schedule. The dispatching rules available can be classified in three sets: priority rules that determine a list of recipes to be sequenced and assigned to specific units, assignment rules that determine which equipment should be used for each batch, and sequencing rules that determine the sequence of batches and the sequence of operations for each unit.

It also has a library containing a variety of heuristic and rigorous optimization algorithms, such as rule-based and genetic algorithms, to determine an initial optimum schedule. The objective function can be specified to optimize for, e.g., use of resources and cost of changeovers. The scheduler generates a first batch sequence based on the present state of the plant, the predicted start and end times of the operations, and the orders of products. It generates a list of control recipes, which are sent to the coordinator, via the DTM, using the Batch Markup Language (BatchML) [World Batch Forum, 2003] format. BatchML provides a set of XML schemas based upon the S88 family of standards. BatchML may be used to design interfaces in control systems as well as the basis for documenting requirements, designs, and actual product and process data.

The time for performing a certain operation may increase over time due to, e.g., fouling or catalyst degeneration, and it may decrease when the heat transfer areas are cleaned or the catalyst is regenerated. Such

time trends can be used to predict future operation times based on the feedback of the real execution times for the process operations from the coordinator. The coordinator also sends back information about the state of the plant, such as equipment availability and exceptions. Large deviations from the original schedule and information about equipment breakdowns may trigger a rescheduling of the batches based on the new information. If the new schedule differs from the recipe being executed in the batch control system it is sent to the process coordinator to replace the current one. This kind of rescheduling is called reactive scheduling. A simplified description of the rescheduling algorithm [Arbiza *et al.*, 2003b] consists of the following steps in the scheduler.

1. Create an initial schedule.
2. Send the schedule to the process coordinator.
3. Receive the actual executed schedule from the process coordinator.
4. Generate a new optimal schedule.
5. If the new schedule significantly differs from the schedule executed by the coordinator go to 2, else go to 3.

The abortion of a batch due to an exception also leads to the generation of a new schedule.

The rescheduling system is configurable and customizable considering the manager objectives. It allows selecting different dispatching rules, optimizers, and objective functions according to the process knowledge. The alternative rescheduling techniques (recalculate a new robust schedule, recalculate a schedule without robustness considerations, reassignment, etc.) are evaluated and the system selects the best suited one according to the objective function being used. Optimization algorithms may be included depending on the interest of the decision maker and the required reaction time.

MOPP publishes the schedules of batches to be processed and subscribes to the actual progress of the scheduled batches published by the coordinator.

Fault Detection System

A fault diagnosis system (FDS), called ExSit-M, has been developed at UPC. It consists of an artificial neural network (ANN) structure together with a fuzzy system in a block-oriented configuration. The FDS combines the adaptive learning diagnostic procedure of the ANN with the transparent deep knowledge representation using a structured form of knowledge-based expert systems. In the rest of this chapter it is assumed that the FDS correctly detects and isolates the faults when they occur.

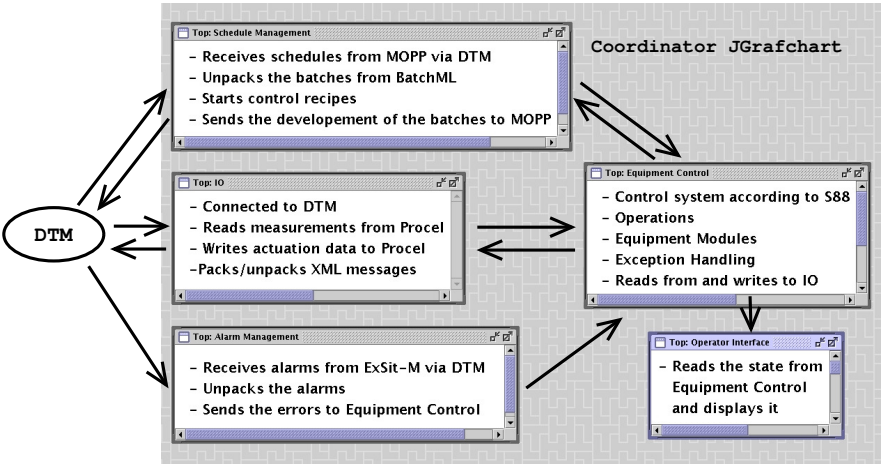

Figure 4.2 Outline of the modules in the coordinator. Schedule/recipe management, operator interface, equipment control, IO, and alarm manager

ExSit-M subscribes to measurement data from the plant and actuator data from the coordinator and it publishes alarms to the DTM when abnormalities are detected.

Coordinator

The coordinator is implemented in JGrafchart. The control system in the coordinator includes management of scheduled batches, recipe execution, unit supervision, alarm propagation, and exception handling and it follows the batch control standard S88. The coordinator consists of a number of different modules in: schedule/recipe management, operator interface, equipment control, IO, and alarm manager, see Fig. 4.2.

The schedule manager unpacks the schedule from BatchML to objects in JGrafchart. Each batch is started as a control recipe in the recipe manager. The unit supervision is based on modeling the state of each equipment object and procedural element using finite state machines as described in Chapter 3. The exception handling takes place at both the recipe and the equipment level.

The coordinator subscribes to the schedules published by MOPP, measurement data from the Procel plant, and alarms from the FDS toolbox. The coordinator publishes actuator data and the actual progress of the recipes.

4.4 Procel

The Department of Chemical Engineering at UPC has developed a batch pilot plant called Procel, see Fig. 4.3. The plant consists of three tanks of glass equipped with agitators, heaters, level sensors and temperature sensors. The tanks are connected in a highly flexible way using pumps, pipes, and magnetic valves, see Fig. 4.4. The plant also has a system of heat exchangers for heating and cooling. Currently these are only used when the plant is running in continuous mode. However, they could be used during the transfer of a batch from one tank to another or during circulation within a unit. The physical plant is located in Barcelona, but there also exist a realistic simulation model of the plant implemented in MATLAB/Simulink. The simulation model has been used to develop the batch control system for the real plant, see [Olsson, 2002].

Procel has been used as a test pilot plant in the research of monitoring, control, on-line fault diagnosis and reactive scheduling of batch processes, see [Ruiz *et al.*, 2001; Cantón *et al.*, 1999] for some of this work. Here the

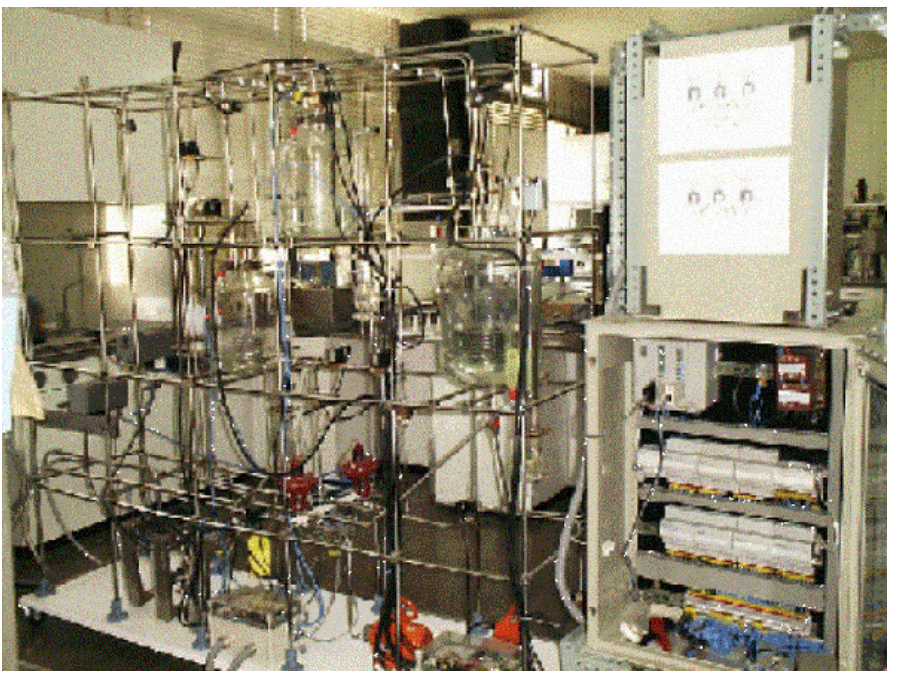

Figure 4.3 Photo of the Procel plant.

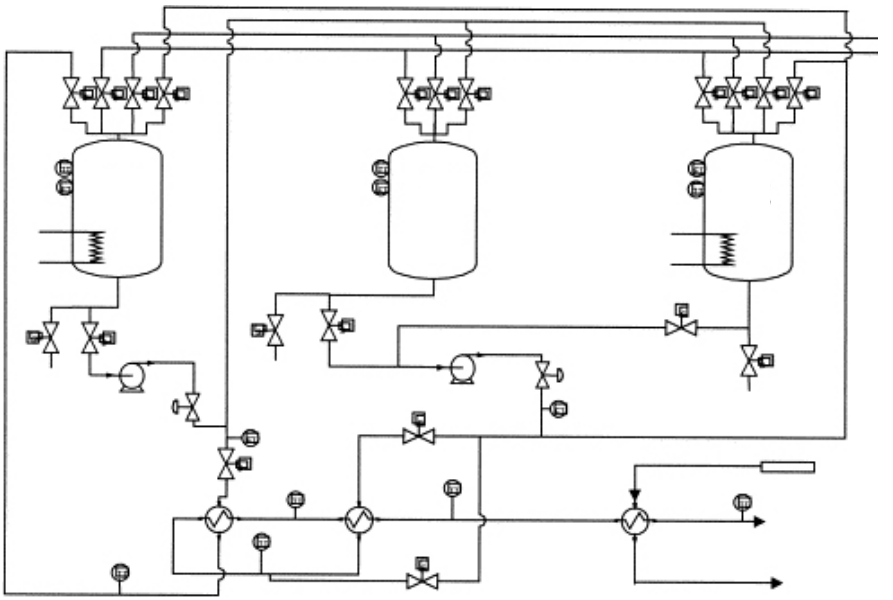

Figure 4.4 A schematic overview of the Procel plant, from [Ruiz *et al.*, 2001].

pilot plant is used as a realistic example to test the exception handling scheme and to integrate the different toolboxes to close the information loop within the plant.

During operation Procel connects to the DTM as a client like the toolboxes. It publishes measurement data and it subscribes to the actuation signals from the coordinator.

Control System

The original control system for the Procel plant is implemented in ABB Sattline [Johanneson, 1994], which is still running underneath the new control system. Sattline is used as a safety net and has several safety and equipment interlocks, which would have been necessary to re-implement otherwise. In this way safety is achieved and the use of the coordinator becomes more flexible. The coordinator sends commands to Sattline, which changes the states of the physical equipment within the process. In the rest of this report Sattline is considered as a part of the Procel plant instead of a part of the control system.

4.5 Integration

The proposed and implemented framework for the integration can be seen in Fig. 4.5. The scheduler sends an initially generated schedule to the coordinator and the FDS, (1). The coordinator starts control recipes according to the schedule and sends control actions to the plant and the FDS, (2). During the execution of the recipes the coordinator sends the actual start and finishing times of the phases and operations of the control recipes to the scheduler and the FDS, (4). The FDS and the exception handling logic in the coordinator detect deviations from the original schedule and the scheduler can update the average duration of the operations/phases in its library based on this data. Procel sends measurements to the coordinator and the FDS to be used for the control and fault detection, (3). If an exception (fault) occurs an alarm is sent from the FDS to the coordinator, (5). The coordinator takes the appropriate control actions depending on the fault. The exception is reported back to the scheduler, (6). Depending both on the duration of the executed schedule and on the severity of the exception the scheduler makes a decision if a new schedule should be sent or if the original schedule should continue. If a batch has to be aborted the batch needs to be re-scheduled and a new schedule is needed.

4.6 Implementation of the Coordinator

In this section the implementation of the coordinator is described. The different parts: management of scheduled batches, recipe execution, unit supervision, alarm propagation, and exception handling are described. The exception handling structure described in Chapter 3 is used in the Procel implementation. The internal model approach where each equipment object is extended with a state machine-based model is used for unit supervision.

Schedule Management

As mentioned above the scheduler publishes a schedule of batches in the DTM as an XML object following the BatchML standard. The coordinator receives the schedule through its subscription to the DTM. It then unpacks the batches one at a time and starts a control recipe for each batch by making a procedure call to the appropriate procedure containing the control recipe. The control recipe is started with the parameters given in the XML object sent from the scheduler. The recipe procedure is implemented as a Grafchart function chart, e.g., according to Fig. 4.6. The recipe waits for its requested start time before it tries to start to execute.

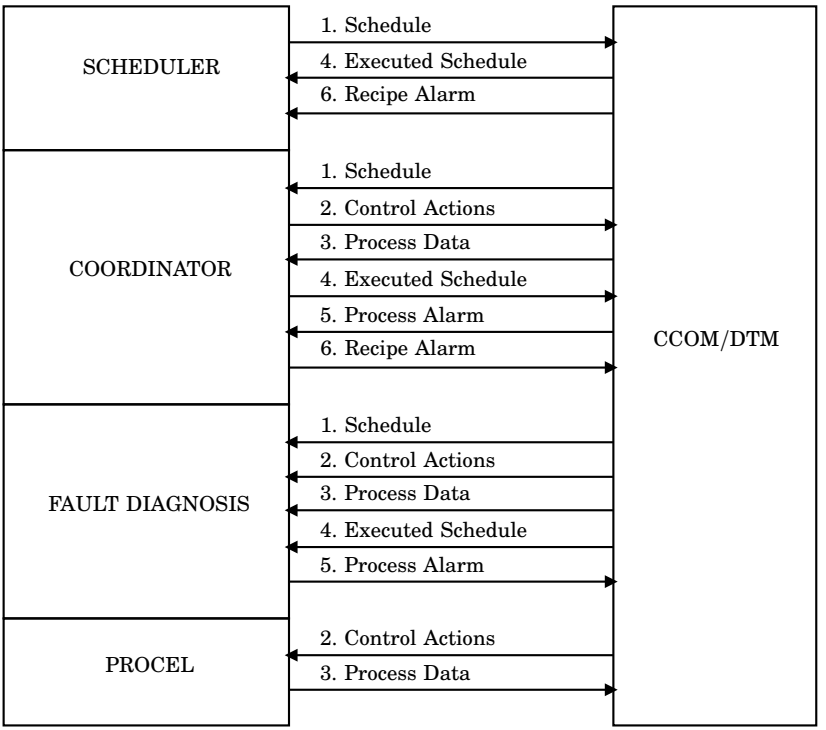

Figure 4.5 Overview of the information flow in the integrated system.

The top level of a schedule contains a list of batches. Each batch contains the following information from the BatchML standard:

- Status
- Recipe ID
- Batch ID
- Product ID
- Order ID
- Batch Priority
- Requested Batch Size
- Actual Batch Size
- A number of Unit Procedures

Each Unit Procedure contains:

- Unit Procedure ID
- Status
- Requested Equipment ID
- Actual Equipment ID
- A number of Operations

Each Operation contains:

- Operation ID
- Status
- Requested Start Time
- Actual Start Time
- Requested End Time
- Actual End Time

The schedule information of the recipe in Fig. 4.6 in BatchML format can be found in Appendix B.

Recipe Management

Several different recipes can be run in the Procel plant. Each recipe is implemented as a procedure in JGrafchart. The implementation of the test recipe in JGrafchart is shown in Fig. 4.6. This is the recipe mainly used during the testing of the integration and it contains the following sequential operations:

- Wait for the start time of the batch according to the schedule.
- Reserve the second unit, `U2`.
- Charge `U2`.
- Hold the content in `U2` for X time units.
- Reserve the first unit, `U1`, and the fourth unit, `U4`.
- Transfer the content in `U2` to `U1` using pump `B2` in `U4`.
- Release `U2` and `U4`.
- Heat the content of `U1`.
- Empty out the content of `U1`.

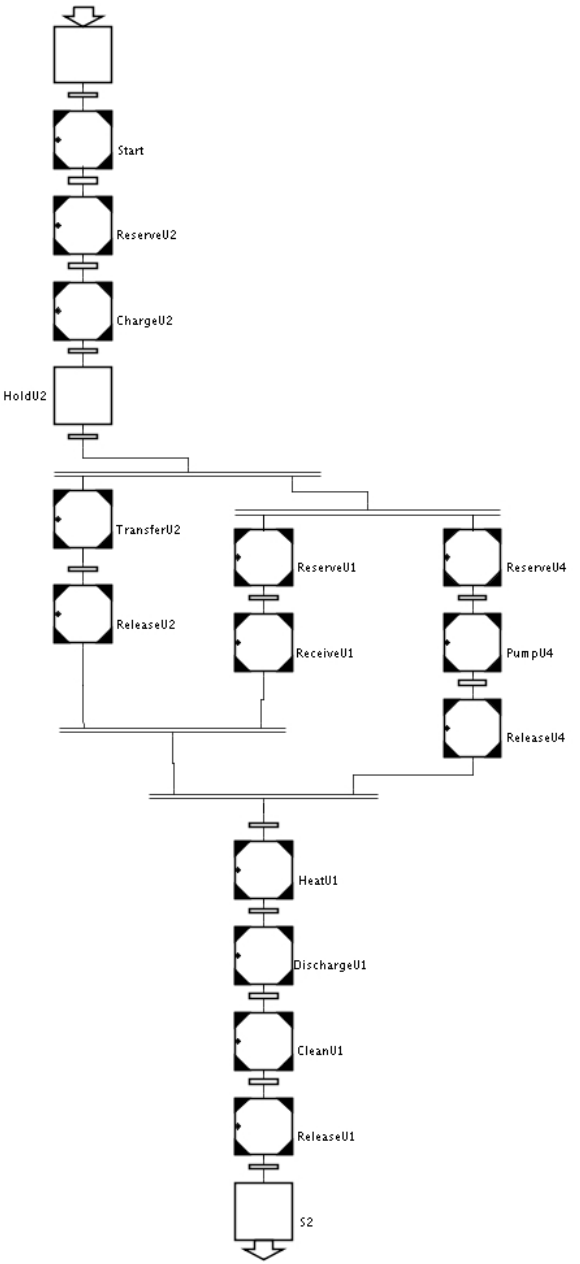

Figure 4.6 The test recipe for the Procel plant implemented in JGrafchart. The first column of procedure steps concern operations in `U2`, the second column concerns operations in `U1`, and the last column concerns operations in `U4`.

- Clean U1.
- Release U1.

Since each batch is a control recipe carrying all the information about the execution of the batch a report can be sent back to the scheduler every time a new phase is started during normal operation. As described earlier, if an exception occurs and a batch has to be canceled this information is sent back to the scheduler. The scheduler is then able to reschedule the batches on-line. The historical data from every batch is stored in the DTM for documentation and optimization of the process and to update the timing trends in the scheduler. The historical database is important to be able to detect frequently occurring problems and to be able to concentrate resources to eliminate these.

Equipment Control

The plant is considered to be a process cell and it is divided into four units according to the physical model in S88. For the operator interface and the names of the equipment, see Fig. 4.8. The first three units, U1–U3, represent the three tanks with the associated equipment and the fourth, U4, consists of auxiliary equipment not part of any of the other units and should maybe not be considered as a unit at all, but as shared resources. The valves and sensors, i.e. flow meters and temperature sensors, used in the heat exchanger network are part of U4. The pump B2 is also in U4 since it is used in different ways during different operations and it is not directly part of any of the other units. Since B2 is used to pump from both U2 and U3 it contains its own operation for pumping.

Each of the units consists of equipment and control modules such as agitators, valves, level, and temperature sensors. In the control system the equipment modules and control modules are stored on sub-workspaces of the unit's workspace. The units also contain the equipment control logic. In the Procel control system implementation the recipe/equipment control separation is on the operation level in S88. This means that the recipe makes a procedure call from a procedure step representing a recipe operation to a procedure representing the corresponding equipment operation belonging to a unit.

The operations, e.g., charge, heat, clean, transfer, and discharge are stored on sub-workspaces located on the unit's sub-workspace according to Fig. 3.15. The equipment/control modules representing the valves, heater, agitator, and sensors are also stored on sub-workspaces located on the unit's sub-workspace. In the Procel implementation the reservation and release of units take place at the recipe level, i.e., the recipe makes a procedure call to the operations ReserveUnit and ReleaseUnit to reserve

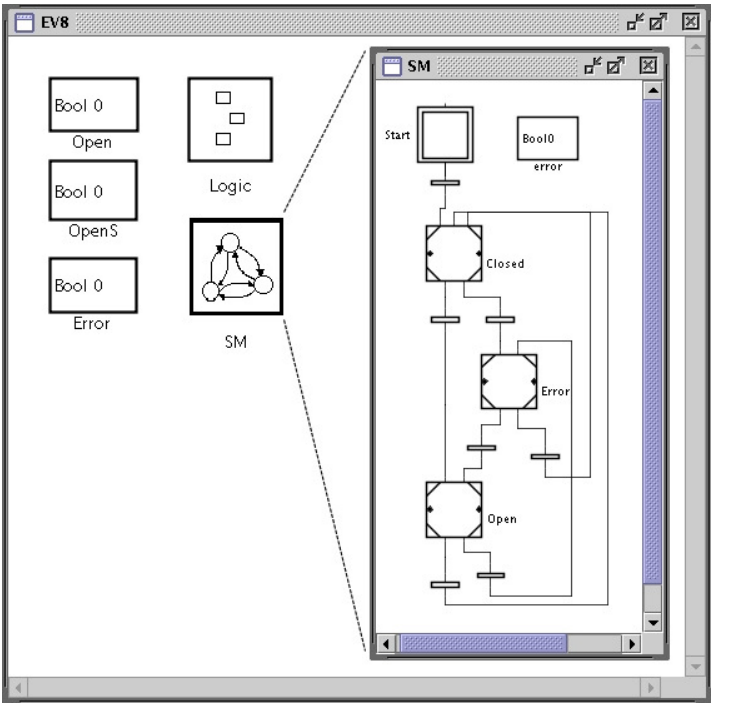

Figure 4.7 Workspace of the equipment module EV8, a valve in U2.

and release the unit, see Fig. 4.6. The complete control system for Procel in JGrafchart consists of 4 units, 39 equipment/control modules, 27 operations/phases, and 111 state-machines.

Unit Supervision

The workspace of a valve contains its state machine and three boolean variables Open, OpenS, and Error, see Fig. 4.7. The Open variable is the current control signal to the valve. The OpenS variable is the current state of the valve according to the measurements from Procel. The Error attribute becomes true if the error detection logic in the FDS detects any error with the valve, e.g., if it fails to respond to an open command within a certain time. On the IO Logic sub-workspace object is a function chart, which reads from and writes to the IO module connected to the DTM.

The equipment-state machines of the units, which are modeling the state of the units, are located on the different unit's workspace. The state machines have the states Available, Reserved, and Error. The state ma-

chines reach the `Error` state if any of the equipment/control modules reach their `Error` state.

Equipment Level Exception Handling

The exception handling for the Procel plant at the equipment level is implemented according to Section 3.3. Checks and control are implemented in the fashion of Fig. 3.17 and the start-state machines as in Fig. 3.18.

Recipe Level Exception Handling

In the Procel plant implementation the step fusion sets are abortive, which allows the transition to immediately fire before the execution of the procedure called from the recipe is finished. The execution of the operation called from the control recipe is aborted and the abortive actions of the procedure step in the control recipe are executed. The transition associated with the specific exception that has occurred becomes true, and the logic for handling the specific exception starts.

Graphical User Interface

The graphical operator interface implemented in the coordinator can be seen in Fig. 4.8. The interface displays the current values of the sensors, the state of the different equipment objects, and alarms. This could be extended with dialogs to make it more interactive, e.g., the graphical interface could link to the workspaces of the different equipment units.

4.7 Information Flow in the Coordinator

The information flow between the modules in the coordinator is different depending on if an exception occurs during the execution of an operation or not. First the execution of an operation in a batch without any exceptions is described and in the next section the execution of a faulty batch is described. During the execution of the operation in both of the scenarios the IO reads the measurements and writes the actuator data in the DTM at a specified sampling rate.

Normal Processing Scenario

The execution of a normal batch without any exceptions is much less complicated than if an exception occurs. The steps of the recipe execution for a normal batch are as follows. The numbers are references to Fig. 4.9.

- When the Scheduler Manager in the Coordinator receives a schedule (1) from the scheduler a control recipe is started in the recipe manager for each batch (2).

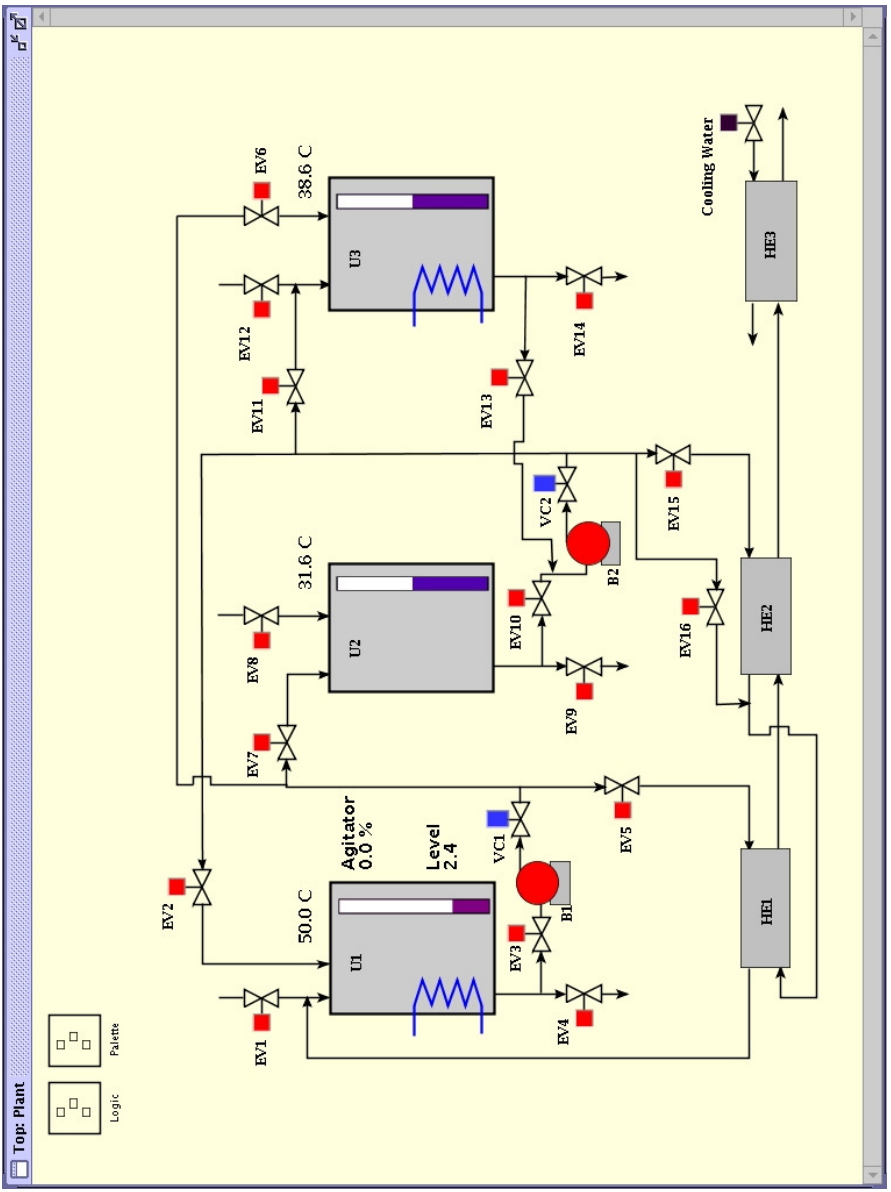

Figure 4.8 Operator interface for the Procel plant in JGrafchart

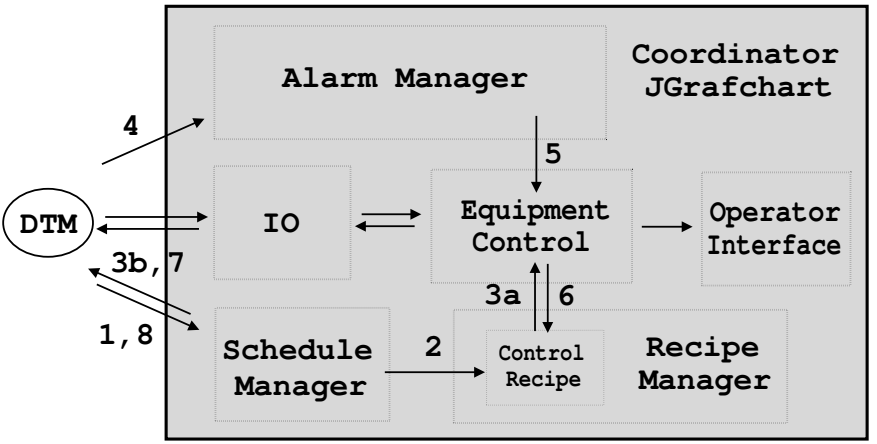

Figure 4.9 Information flow to/from and within the Coordinator during the execution of a normal and a faulty batch.

- The normal execution of the control recipe takes place through procedure calls from the control recipe to the Equipment Control (3*a*), e.g. `Fill`, `Transfer`, and `Heat`. The execution of the batch is continuously reported back to the scheduler MOPP (3*b*) by posting data in the DTM about the actual start and end times of the operations.

Exception Handling Scenario

The recipe in Fig. 4.6 is used as an example of the exception handling. The exception chosen for this test is that the heater `R1` in unit `U1` is not responding to a command from the `Heat` operation. The steps of the recipe execution and the exception handling for a faulty batch are as follows. The numbers are references to Fig. 4.9.

- The Schedule Manager receives a schedule from MOPP (1) and starts a control recipe (2) for each of the scheduled batches. The recipe calls operations in the Equipment Control (3*a*) according to normal execution, described above, until it reaches the step where the `Heat` operation is called. The evolution of the batch is sent back to MOPP via the DTM (3*b*).
- The state of `U1` is checked to be consistent with the start of the `Heat` operation.
- The procedure-state machine of the `Heat` operation goes from the `Idle` state to the `Running` state.

- The Heat operation sends a command to the R1 equipment module and sets the control signal to 50% of the capacity.
- The control signal of R1 is written to the IO, which sends the control signal to Procel via the DTM.
- When R1 is not heating the toolbox for Fault Detection and Diagnosis, ExSit-M, detects this and publishes the alarm in DTM (4). The alarm is unpacked by the Alarm Manager and sent to the Equipment Control (5). This causes the state machine of R1 to go to the Error state.
- The Error state propagates to the state machine of the unit and the recipe level exception handling logic detects it (6).
- The Error Exit, in the recipe level exception handling, corresponding to that the heater is not responding is fired.
- The Heat operation is aborted and its procedural state machine goes to the Held state. The abortion of the recipe is sent to MOPP (7).
- The exception handling operation for the heater tells the operator that the heater is not responding by displaying the alarm in the Operator Interface.
- The heater R1 is repaired and the state machine of R1 is reset from the Error state.
- If no actions were taken that prevents the operations from being restarted, the recipe level exception handling makes a rollback and the exception detection and the Heat operation are restarted. If on the other hand the recipe cannot be restarted appropriate actions must be taken, e.g. the reactants must be disposed in a safe way. The recipe is aborted and the scheduler makes a re-scheduling and sends a new schedule to the Schedule Manager to make a new batch (8).
- The procedure-state machine of the Heat operation goes from the Held state to the Running state via the Restarting state and the heating of the batch continues.
- When the specified temperature is reached the Heat operation sends a command to R1 to stop heating.
- The procedural state of the Heat operation goes to the Idle state when R1 has stopped heating.
- The recipe continues according to normal operation.

4.8 Summary

In this chapter the development of a batch control system for the Procel pilot plant at UPC is described. The control system is an integration of scheduling, fault detection and diagnosis, and a recipe coordinator including exception handling. The different parts are integrated using the CHEM Communication Manager (CCOM) and a Data Manager (DTM).

A small number of exceptions have successfully been implemented to test the structure of the exception handling in the control system. To find solutions for different scenarios and plants the proposed approaches should be further tested and implemented in larger batch control systems. The suggested methods need to be tested on industrial plants to determine the practical aspects of the proposed structure.

During the development of the control system on-line tests was performed by running the coordinator in JGrafchart at the Department of Automatic Control in Lund while the other parts was running at UPC in Barcelona. The developers communicated with different chat programs during the tests. For a demonstration at a CHEM meeting in Lille in France, the development team from UPC brought laptops to run their different parts there. The control system was demonstrated on the real process on-line from Lille together with the coordinator running in Lund and the Procel plant of course in Barcelona.

5

Process Fault Detection and Isolation

5.1 Introduction

A process fault can be any kind of malfunction in a dynamic system or plant, which leads to unacceptable performance such as personnel injuries or bad product quality. Faults may occur either in the sensors, the actuators, the control system, or any other components of the process. The increasing complexity of the control systems in modern plants demands a higher level of fault tolerance and an efficient fault detection and isolation is one part to achieve this. The three steps: *detection*, i.e., detect that something abnormal has occurred in the process, *isolation*, i.e., isolate and find the source for the malfunction, and *correction*, i.e., the actions taken to take the process back to a normal state are here summarized in the term *diagnosis*. Automation of fault detection and isolation for dynamic systems is a large research area where a lot of work have been performed over the last decades. Some good surveys and reviews that summarize this work are [Frank, 1990; Venkatasubramanian *et al.*, 2003b; Venkatasubramanian *et al.*, 2003a; Venkatasubramanian *et al.*, 2003c], where the aim is to give a systematic and comparative study of various diagnostic methods from different perspectives.

The product quality in a chemical plant is maintained by controlling the state of the process. Often the quality variables cannot be directly measured and the quality is indirectly controlled by other variables. When the operating condition varies outside the design limits, not only product quality is at risk, but also the personnel and the environment may be affected in a serious way. Diagnosis of process faults is difficult even for well trained and motivated operators since the faults may be sudden and,

hopefully, infrequent. Since the time from the occurrence of a fault to the detection of the fault and the time to when the actions are taken to correct the fault are critical factors, hesitation as well as erroneous actions may lead to serious situations. Automation of fault detection and isolation, if properly designed and implemented, is a support to operators in stressful situations and could reduce the risk for the personnel as well as save money.

Faults may influence the process in a multiplicative or an additive manner. Faults can have different types of causes such as parameter changes, e.g., input concentration and heat transfer coefficients, or hardware failures, e.g., breakdown of sensors, actuators, and controllers. They can be abrupt, e.g., sudden stop of a pump, or incipient, e.g., fouling slowly developing in a heat exchanger. An overview of how noise, disturbances, and failures can enter in a feedback-controlled dynamic system can be found in Fig. 5.1.

When comparing different types of diagnosis methods it is good to have well defined criteria to be able to list the pros and cons. Some desirable characteristics of an automated fault diagnosis system are [Venkatasubramanian *et al.*, 2003b]:

- Fast detection.
- Good isolation properties, determine which fault has occurred.
- Robust to uncertainties and noise.

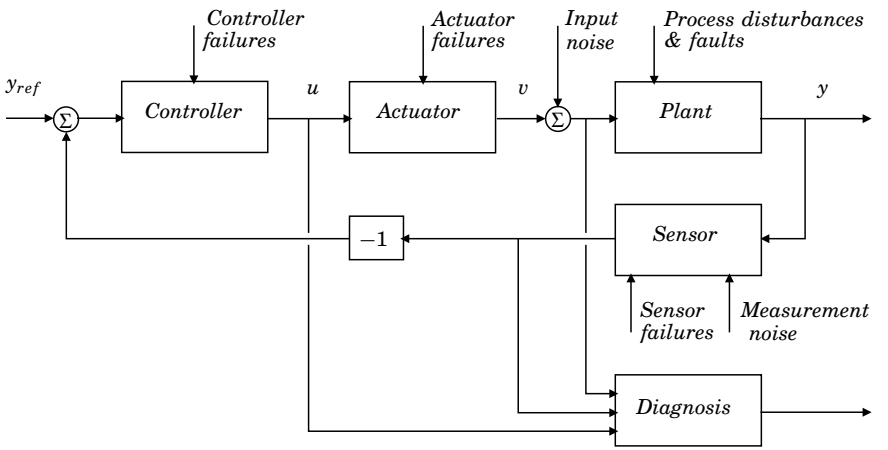

Figure 5.1 Overview of noise, disturbances, and failures in a feedback-controlled dynamic system.

- Adaptation to changes of the process design and operation.
- Low development and maintenance cost.
- Ability to identify multiple faults occurring at the same time.

Different methods have drawbacks as well as advantages compared to others. Often there is a tradeoff between the criteria, e.g., fast detection may make the system sensitive to noise.

In the review paper [Venkatasubramanian *et al.*, 2003b] the methods are divided into three categories, quantitative model-based, qualitative model-based, and process history-based methods. A summary of these three categories is given here.

5.2 Quantitative Model-Based Methods

Models of dynamical systems are used for different purposes, e.g., simulations and control design. A very general description of a model is [Minsky, 1965]

> "To an observer B, an object A^* is a model of an object A to the extent that B can use A^* to answer questions that interest him about A."

Different models can capture different type of information, e.g., structure and topology: 'Which components are present?', 'How are they related?', geographical properties: 'Where is a component located?', 'How large is it?', behavior: 'What do the signals look like?', 'What is the value of the variable x?', and means and ends, i.e., functions and goals: 'How is this task performed?', 'What does this component do?'. This section deals with behavioral models, i.e., dynamic mathematical models, and the use of these for generation of residuals to conclude if a process is in a normal state.

One way of generating residuals for a process is by duplicating the instrumentation and sensors of the process. The residual is then calculated as the difference between the sensors. This way one can conclude that, e.g., one of the level sensors is malfunctioning if the two, or more, sensors show very different values or if one of the sensors shows a much lower signal to noise ratio. This is called *physical redundancy*. Another way of generating residuals is to model the process and compare the output from the model with the measurements from the real process. This is called *analytical redundancy*. The greatest gain with analytical redundancy is that there is no need of adding any new equipment, which may be very expensive. Mathematical models are used to calculate the residuals using

the control signal as input to the model and comparing the outputs from the model and the process to form residuals. The wish is that the residuals should be close to zero when no fault is present and if a fault occurs the model should no longer be valid with large residuals as a result. The aim is that the residuals should be robust to unknown inputs like modeling uncertainties and measurement noise.

In quantitative model-based fault detection and isolation explicit mathematical models are used for the generation of the residuals. The models are either developed by a first-principles or a black-box model is identified from experiments. Most of the methods use linear discrete time black-box models, such as input-output and state-space models, due to the fact that first-principles models are complex and that chemical processes are often non-linear, especially batch processes. A first-principles model is based on physical understanding of the process. Models of chemical processes are usually based on heat balance equations and mass balances with low-order reaction kinetics. In a first-principles model the parameter have physical meaning while in a black-box model this is often not the case.

An overview of the generalized structure of a fault diagnosis system based on the qualitative model-based methods described in this section [Isermann, 1984] can be found in Fig. 5.2.

Parity Relations

The most common black-box model is the linear discrete time state space model, often identified from designed experiments to ensure persistent excitation. Such a model, with no faults, disturbances, or noise, is given by

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma u(k) \\ \hat{y}(k) &= Cx(k) + Du(k) \end{aligned} \tag{5.1}$$

where $u(k)$ is the input, $\hat{y}(k)$ is the model output, and $x(k)$ is the state vector. Φ, Γ, C, and D are matrices of appropriate size. Using the pulse-transfer operator $H(q)$ the system can be written as

$$\hat{y}(k) = \left(C\left(qI - \Phi\right)^{-1}\Gamma + D \right) u(k) = H(q)u(k) = \frac{B(q)}{A(q)}u(k) \tag{5.2}$$

where q is the forward-shift operator. The primary residuals, $\tilde{r}(k)$, are then given by

$$\tilde{r}(k) = y(k) - \hat{y}(k) = \begin{bmatrix} I & -H(q) \end{bmatrix} \begin{bmatrix} y(k) \\ u(k) \end{bmatrix} \tag{5.3}$$

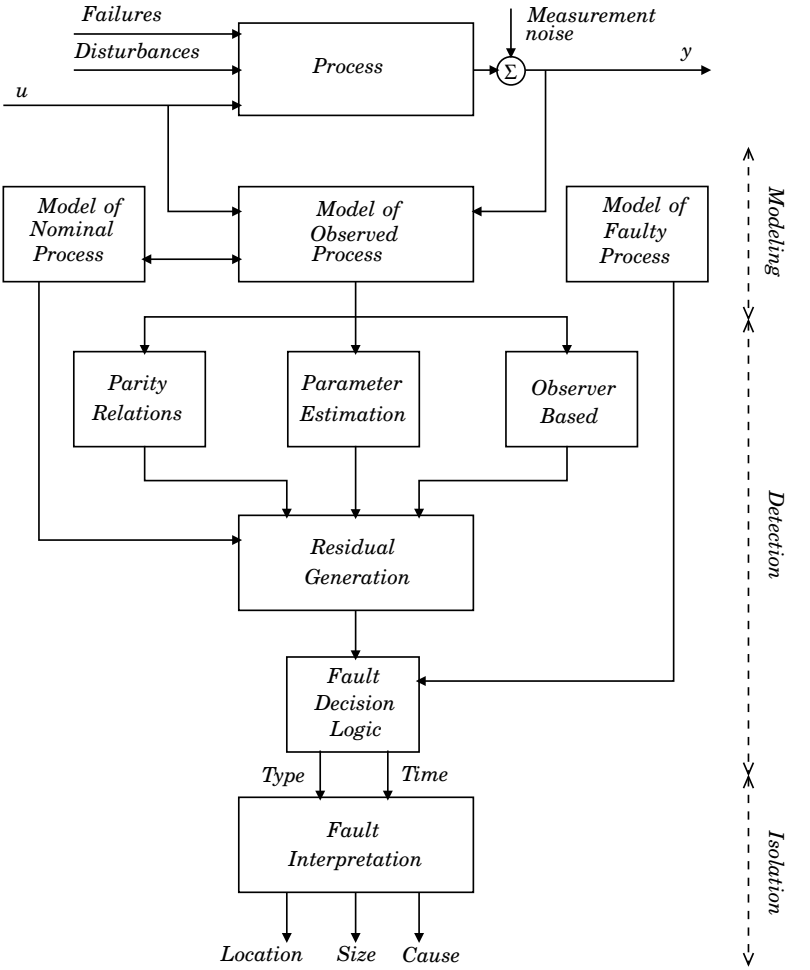

Figure 5.2 Overview of the generalized structure of a fault diagnosis system based on the qualitative model-based methods [Isermann, 1984].

Multiplicative faults, or parameter faults, can be modeled by adding discrepancy matrices $\Delta A(q)$ and $\Delta B(q)$ according to

$$(A(q) + \Delta A(q))\,\hat{y}(k) = (B(q) + \Delta B(q))\,u(k) \tag{5.4}$$

Since A and B are functions of the v process parameters $\Theta = [\Theta_1\ \Theta_2\ \ldots\ \Theta_v]^T$

the partial derivatives become

$$\begin{aligned} Q_j(q) &= \frac{\partial A(q,\Theta)}{\partial \Theta_j} \\ R_j(q) &= \frac{\partial B(q,\Theta)}{\partial \Theta_j} \end{aligned} \tag{5.5}$$

and thus

$$\begin{aligned} \Delta A(q) &= \sum_{j=1}^{v} Q_j(q)\Delta\Theta_j \\ \Delta B(q) &= \sum_{j=1}^{v} R_j(q)\Delta\Theta_j \end{aligned} \tag{5.6}$$

By writing

$$\begin{aligned} e_j(k) &= Q_j(q)u(k) - R_j(q)\hat{y}(k) \\ E(k) &= [e_1(k)\ e_2(k)\ \ldots\ e_v(k)] \end{aligned} \tag{5.7}$$

Eq. (5.4) becomes

$$\begin{aligned} A(q)\hat{y}(k) &= B(q)u(k) + (\Delta B(q)u(k) - \Delta A(q)\hat{y}(k)) \\ &= B(q)u(k) + E(k)\Delta\Theta \end{aligned} \tag{5.8}$$

The additive faults can be added to this and the model becomes

$$A(q)\hat{y}(k) = B(q)u(k) + D(q)d(k) + F(q)f(k) + E(k)\Delta\Theta \tag{5.9}$$

where $f(k)$ are the additive faults and $d(k)$ is a disturbance. This approach is what is called parity relations [Gertler and Singer, 1990; Gertler and Kunwer, 1995; Gertler, 1998].

Isolation To isolate a fault, i.e., find the root cause, and take the appropriate actions the generated residuals are analyzed using fault decision logic. Eq. (5.3) is how the primary residuals are calculated. To improve the isolation of faults the primary residuals can be transformed using $W(q)$.

$$r(k) = W(q)\tilde{r}(k) = W(q)\left[y(k) - H(q)u(k)\right] \tag{5.10}$$

where $W(q)$ needs to be a stable transfer function, which also stabilizes $H(q)$. This way the model does not have to be stable for the residual generation to work.

Two different ways of choosing $W(q)$ that gives *structured residuals* and *directional residuals* can be found in [Gertler and Kunwer, 1995]. In the former for a specific fault only a specific subset of the residuals become nonzero and in the latter the residual vector will have a specific direction associated with the occurring fault.

Non-Linear Processes If the process is non-linear the linear discrete time model can only be used if the process is operating around a linearization point. Therefore this approach is not easy to use for batch processes. Some extensions of analytical redundancy methods to non-linear systems have been developed. The proposed methods rely on theory for polynomial differential-algebraic equations. The fact that the most commonly used functions, e.g., trigonometric, exponential, and logarithmic functions, can be written on polynomial form makes this possible. These methods are based on elimination theory, see [Staroswiecki and Comtet-Varga, 2001; Isidori *et al.*, 2001; Frisk, 2001].

Parameter Estimation-Based Methods

The fundamental idea of fault diagnosis based on parameter estimation [Isermann, 1989; Isermann, 1993] is that many faults appear as changes in the process coefficients p, e.g., heat transfer coefficients, ambient temperature, and reaction coefficients. In the process model the coefficients are contained in the parameters θ, which are assumed to be constant or time-dependent. The parameters θ are often combinations of several of the real process coefficients p. For example, if the model is on the form of a linear differential equation

$$a_0 y(t) + a_1 \dot{y}(t) + \ldots + y^{(n)}(t) = b_0 u(t) + b_1 \dot{u}(t) + \ldots + b_m u^{(m)}(t) \quad (5.11)$$

then the process parameters are given by

$$\theta^T = [a_0 \; a_1 \ldots \; a_{n-1} \; b_0 \; b_1 \; \ldots \; b_m] \quad (5.12)$$

A procedure for using parameter estimation for fault diagnosis may consist of the following [Isermann, 1989]. First a model is developed, e.g., on the form

$$y(t) = f(u(t), \theta) \quad (5.13)$$

where y is the outputs, u are the inputs, and $\theta = [\theta_1 \ldots \theta_J]$. Then a relationship between the parameters θ and the coefficients $p = [p_1 \ldots p_I]$ is determined

$$\theta = g(p) \quad (5.14)$$

The parameters are estimated from measured inputs u and outputs y during operation of the process using any suitable estimation method, e.g., recursive least squares, RLS, which is further described in Chapter 10. As in all estimation techniques it is important that the system is persistently excited to be able to get correct estimates. The physical coefficients are then calculated from the estimated parameters

$$p = g^{-1}(\theta) \quad (5.15)$$

It is not always possible to determine all coefficients from the parameters and sometimes it is enough to only monitor the parameters θ. The estimated coefficients p are compared to nominal values determined during the modeling of the process, and if p is deviating from the nominal values a fault is signaled. Fault signatures, i.e., how a certain fault affects the coefficients are used to isolate the fault.

Observer-Based Methods

The perhaps most commonly used method for generating the residuals is by using observers, or Kalman filters, to estimate the state of the process [Patton *et al.*, 1989]. The method is based on using a set of observers, where each observer is sensitive to a subset of the possible faults while insensitive to the rest of the faults and to the unknown inputs. If there are no faults all the observers should track the process perfectly and the residuals should be close to zero. When a fault of occurs only the observers that are sensitive to the specific fault should give a large residual while the other remain small. The pattern of the residuals is used to isolate the fault.

Let the true system be given by a discrete linear-time invariant state-space model

$$\begin{aligned} x(k+1) &= \Phi x(k) + \Gamma u(k) + Ed(k) + Gf(k) \\ y(k) &= Cx(k) + Du(k) + Fd(k) + Hf(k) \end{aligned} \tag{5.16}$$

where $Ed(k)$ and $Fd(k)$ models the input and output disturbances, $Gf(k)$ models actuator and process faults, and $Hf(k)$ models sensor faults. The initial state $x(0) = x_0$. An observer for this process is given by [Åström and Wittenmark, 1997]

$$\begin{aligned} \hat{x}(k+1 \mid k) &= \Phi\hat{x}(k \mid k-1) + \Gamma u(k) + K\big(y(k) - \hat{y}(k \mid k-1)\big) \\ \hat{y}(k \mid k-1) &= C\hat{x}(k \mid k-1) + Du(k) \end{aligned} \tag{5.17}$$

where $\hat{x}$ and $\hat{y}(k)$ are the estimated state and output, and K is the observer gain. The state estimation error is given by $\tilde{x} = x - \hat{x}$ and the output estimation error is $e = y - \hat{y}$, with the dynamics

$$\begin{aligned} \tilde{x}(k+1 \mid k) &= (\Phi - KC)\,\tilde{x}(k \mid k-1) + (E - KF)\,d(k) + (G - KH)\,f(k) \\ e(k) &= C\tilde{x}(k \mid k-1) + Fd(k) + Hf(k) \end{aligned} \tag{5.18}$$

Since e is a function of d and f and not of u it can be used as a residual for detection of faults. The design of the fault detection system is to choose a proper observer gain. The goal is that the faults, f, should be decoupled

from each other and that the residual is invariant to the unknown inputs, i.e., disturbances d.

The Kalman filter can also be augmented to estimate parameters of the process. To be able to estimate states and parameters it is necessary that the system is fully observable.

Non-Linear Processes For non-linear systems an extended Kalman filter, EKF, can be used. The EKF is based on linearization of the non-linear system around a trajectory such that standard Kalman filter theory can be applied. The EKF is computationally intensive compared to linear Kalman filtering. Terms neglected in the linearization may be relatively large and can introduce large errors, which may lead to sub-optimal performance of the filter. The derivation of the Jacobian matrices are often nontrivial and can lead to significant implementation difficulties. The EKF will be further discussed in Chapter 10.

Simultaneous Design of Controller and Fault Detection

In [Tyler and Morari, 1994] an approach for designing the controller and an observer-based fault detection is presented. It has been further developed and implemented in, e.g., [Nett *et al.*, 1988; Niemann and Stoustrup, 1997; Åkesson, 1997]. In this framework it is shown that uncertainties in the system may result in a trade off between control performance and diagnostic performance.

In the method the system is rewritten so that the controller and the fault detection for the uncertain linear system can be designed using standard tools for robust control, such as H_∞ or μ-synthesis, see, e.g., [Zhou, 1998].

Diagnostic Model Processor

The diagnostic model processor DMP [Petti, 1992] uses model predictions and measurements to find a set of violated assumptions, i.e., likely faults. DMP is based on that during normal operation the predictions from the model equations are close to the measurements. The model equations are rewritten to residual form and upper and lower limits for the residuals are assigned to define normal operation. A thorough presentation of DMP and its application to batch processes is found in Chapter 8.

5.3 Qualitative Model-Based Methods

In the quantitative model-based approaches above the knowledge about the system is in the form of dynamic relationships between the inputs

and the outputs of the system. In qualitative model-based approaches the relationships are in the form of qualitative functions or relations describing the process. The functions can describe if a process variable, e.g., is *high*, *low*, *increasing*, or *decreasing*.

One method in this category is a knowledge-based expert system, which contains a knowledge base, usually built with a set of if-then-else rules, and an inference engine that searches through the knowledge base to conclude which events have caused the current state. An example of a rule can be if the control signal to a valve is *Open* and there is no flow then the valve is stuck is the closed position. Knowledge bases often lack understanding of the physics of the process and cannot handle events that are not part of the knowledge base.

Another method in this category is signed digraphs SDG, first used for fault diagnosis in [Iri *et al.*, 1979]. An SDG is a set of nodes representing variables and directed arcs that describe cause-effect relations. A + or − on the arcs represents the gains between two variables (nodes). A gain of +1 indicates that the two variables vary in the same direction. The main usage of digraphs is isolation. Faults are associated with, e.g., the states *High* or *Low* of a variable or introduced as separate nodes with arcs to other nodes describing the influence of the faults. Isolation is performed by locating which fault node that has given rise to the current state usually assuming only a single root cause. An SDG model of a simple tank from [Nilsson *et al.*, 1992] with in-flow, F_{in}, out-flow, F_{out}, and the level in the tank, L, as variables can be seen in Fig. 5.3. Three fault nodes are also introduced representing blockage of the in-flow and out-flow, and leakage in the tank. SDGs have been used in numerous ways for fault diagnosis. The reader is referred to [Venkatasubramanian *et al.*, 2003a] for more information of these and references.

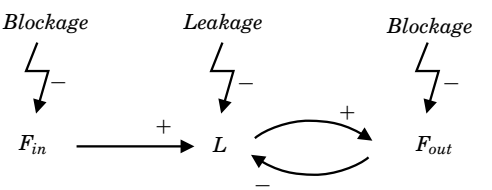

Figure 5.3 An SDG model of a tank with three possible faults.

5.4 Function Based Methods

Multilevel Flow Modeling

Multilevel flow modeling, MFM [Lind, 1990; Larsson, 1992; Larsson *et al.*, 2004], is a technique for modeling means-end models. An MFM model is a representation of a system's functions in terms of what should be done, how it should be done, and with what it should be done. The system is decomposed in two directions, in the first the system is decomposed into subsystems, while in the second direction the goals of the system are related to the functions for achieving these goals. MFM can be used to create knowledge data bases, human-machine interfaces, and as a design tool for both process and control system design.

Fault Tree Analysis

Fault tree analysis, FTA, consists of block diagrams that give a graphical picture of possible means in which a fault could be generated in a design, process, or product. FTA gets its names from the graphic construct used to guide the process to arrive at the cause of a fault. The branching structure resembles the structure of a tree or root system. FTA is inherently a top-down search. It starts with fault, then works its way down through all the subsystems, components and conditions of the system which could contribute to this fault. The tree structure is built often built up by logical gates, i.e, `And` and `Or` gates. More than one hypothetical solution usually results when a complex system is analyzed. FTA is defined in the standard IEC 61025 [IEC, 1990].

Failure Mode and Effects Analysis

Another commonly used technique for risk analysis is usually identified by the name failure mode and effects analysis, FMEA [Stamatis, 2003]. The FMEA technique considers each item that may comprise the total system. Analysis is made of all ways that each component or subsystem might fail. Each of these potential faults is then ranked. The ranking process takes into account three separate aspects of each fault. One ranking is assigned with regard to the relative probability that the particular fault will occur. The fault is also ranked for the relative severity of its worst potential resulting outcome regarding safety and functionality of the system. The third relative ranking is for the probability that the failure mode will be detected and/or corrected by the applicable controls. One of the most powerful aspects of FMEA is the assignment of these relative measures of occurrence, severity and detection/correction. The three numeric rankings are multiplied together for each failure mode to provide an overall relative

risk factor for the subject failure mode. With this relative risk factor the faults that are most likely to cause safety, reliability or quality problems can be identified. FMEA is defined in the standard IEC 60812 [IEC, 1985]

Hazard and Operability Studies

A hazard and operability, Hazop, study identifies hazards and operability problems. It involves investigating how the plant might deviate from the design intent. The prime objective for the Hazop study is problem identification. If during the process of identifying problems, a solution becomes apparent, it is recorded as part of the Hazop result.

Hazop is based on that several experts with different backgrounds and experience of the plant interact and identify problems working together. The Hazop concept is to review the plant in a series of meetings, during which a multidisciplinary team methodically "brainstorms" the plant design, following the structure provided by the guide words and the team leader's experience.

In the Hazop study the plant schematic be divided into study nodes that are addressed with the guide words. The guide words are simple words which are used to qualify or quantify the intention of the equipment or process being considered in order to guide and stimulate the brainstorming process and so discover deviations. Some often used guide words are: `No`, `Less`, `More`, and `Other Than`. Each guide word is applied to the study node which is being examined to figure out what would happen in the node if a fault associated with the guide word occurs. During the meeting a Hazop form is filled out where the identified problems and possible solutions are entered. Hazop is defined in the standard IEC 61882 [IEC, 2001b].

5.5 Process History-Based Methods

In contrast to the methods described above in process history-based methods no knowledge about the process is needed. Only measurements from previously successful operation is used. This data is used to build a diagnostic system. The most used methods are statistical process control, SPC, and multivariate statistical process control, MSPC, based on principal component analysis, PCA. Statistical methods deal with non-deterministic systems. The measurements are considered to be statistical time series, where the observations have probability distributions. The distributions are assumed to be changing when the system is out of control. Statistical methods are treated further in Chapter 6.

6

Multivariate Statistical Methods for Batch Process Monitoring

6.1 Introduction

This chapter contains a description of multivariate statistical methods for process monitoring and their various extensions for handling data from batch processes. Multivariate statistical methods have been developed in different fields of science, e.g., psychology, biology, chemistry, and mathematics. There exist a number of alternative approaches of multivariate statistical methods for process monitoring, e.g., methods based on principal component analysis, PCA, and methods based on partial least squares techniques, PLS. Most of the methods described in this chapter are developed from PCA. It is shown how these different methods are related and when they are equivalent.

There exist some good tutorials and overviews on multivariate statistical methods for process monitoring, e.g., [Kourti and MacGregor, 1995; Qin, 2003; Cinar and Undey, 1999; Bro, 1997; Andersson and Bro, 2000; Piovoso and Hoo (*Eds.*), 2002]. The following notation will be used:

i:	batch number	x:	vector
I:	number of batches	X:	matrix
j:	variable number	$\underline{X}$:	higher-order tensor
J:	number of variables	x_{jk}:	jkth element in a matrix
k:	sample number	x_j:	jth element in a vector or
K:	number of samples		jth vector in a matrix
d:	time lag		

6.2 Principal Component Analysis

The basis for many of the methods described in this chapter is principal component analysis, PCA. PCA tries to explain the covariance structure of a set of variables by finding a small number of linear combinations of the variables. PCA linearly transforms an original set of correlated variables into an often substantially smaller set of uncorrelated variables, called principal components, that represent most of the information in the original set of variables. The multivariate technique of principal component analysis was first described by Pearson in [Pearson, 1901]. A first description of practical computing methods for PCA came from Hotelling [Hotelling, 1933].

PCA assumes that the data is stored in a two dimensional matrix $X \in \mathbb{R}^{K \times J}$

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1J} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2J} \\ \vdots & \vdots & & \vdots & & \vdots \\ x_{k1} & x_{k2} & \dots & x_{kj} & \dots & x_{kJ} \\ \vdots & \vdots & & \vdots & & \vdots \\ x_{K1} & x_{K2} & \dots & x_{Kj} & \dots & x_{KJ} \end{bmatrix} \tag{6.1}$$

where J is the number of variables and K is the number of observations. X is then decomposed into scores T and loadings P according to

$$X = TP^T = t_1 p_1^T + t_2 p_2^T + \ldots + t_R p_R^T + E_R = \sum_{r=1}^{R} t_r p_r^T + E_R \tag{6.2}$$

where R is the number of principal components selected for the model, $t_r \in \mathbb{R}^K$, $p_r \in \mathbb{R}^J$, and $E_R \in \mathbb{R}^{K \times J}$ is the remaining error matrix. The first principal component is the combination of the original variables that explains the greatest amount of variation in X. The second principal component defines the next largest amount of variation and is independent (orthogonal) to the first principal component and so on.

Algorithms

The two major algorithms for performing PCA are singular value decomposition, SVD, and nonlinear iterative partial least squares, NIPALS. The principal components can be calculated directly or, more commonly, after different centering and scaling operations on the data matrix X according to

- Mean centering by subtracting the mean of the columns, i.e., $X_j - \overline{X}_j$.
- Mean centering of each column and scaling by dividing each column by its standard deviation, i.e., $\left(X_j - \overline{X}_j\right)/s_j$.

where $\overline{X}_j$ is a vector with the mean of the jth column, $\bar{x}_j$, duplicated at each entry to have the same dimension as X.

$$\bar{x}_j = \frac{1}{K}\sum_{k=1}^{K} x_{kj} \tag{6.3}$$

and the standard deviation of the jth column is the square root of the variance of the jth column given by

$$s_j = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(x_{kj} - \bar{x}_j\right)^2} \tag{6.4}$$

The second way of preprocessing, where the data is scaled by the standard deviation of each column and the columns are mean centered is commonly called auto-scaling. This scaling gives the original variables equal weight in the analysis. It is often the recommended scaling when no information is available of the importance of the variables. If a variable is known to have a greater impact this variable may be given a larger weight in the analysis. For example, the signal to noise ratio in different variables may be a basis for weighting the variables. Variables describing the same phenomenon can be blocked together and the blocks are scaled separately. Depending on the data in X any of the methods above may give the best result. A discussion on the effects of centering and scaling can be found in [Bro and Smilde, 2003].

To illustrate what happens in PCA when different scalings are used two dependent random variables are simulated for 500 samples ($J = 2$ and $K = 500$). The original data set is plotted in Fig. 6.1. The center of the data is clearly not at the origin and thus the data need to be mean centered. The data has greater variation in one direction than the other. One can choose to scale the data to unit variance (auto-scaling) in the original variables or not. If the data is only mean centered the plot will look like in Fig. 6.2 (a), and if the data is auto-scaled like in Fig. 6.2 (b). A clear difference in the directions of the principal components, given by the orthogonal lines in the plot, can be seen when the two plots are compared. This makes it important to use the same scaling on any new observations as the one used for computing the principal components. If the data is projected onto the space spanned by the principal components the plots look like in Fig. 6.3. In all the figures the sample number 201 is labeled for comparison.

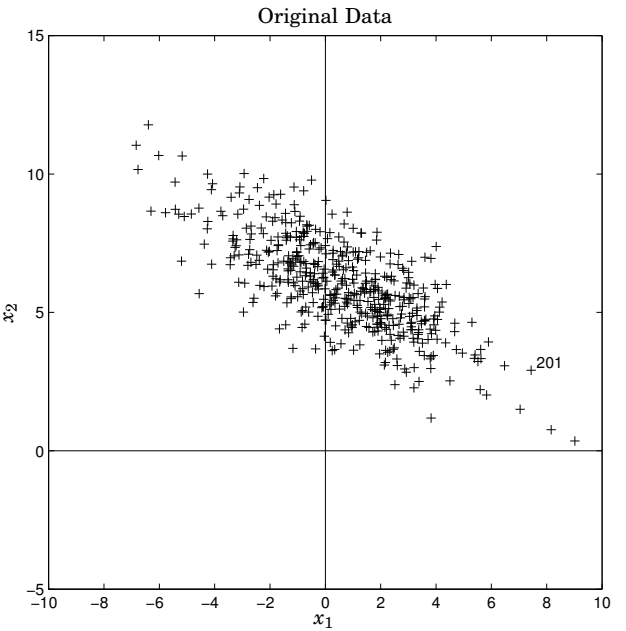

Figure 6.1 Original data used in the PCA example.

Singular Value Decomposition In singular value decomposition, SVD, the matrix X is decomposed according to Eq. 6.5 in Def. 6.2, where U and V are unitary matrices and Σ is a diagonal matrix with the singular values of X on the diagonal ordered from the largest to the smallest, see [Golub and Van Loan, 1996].

DEFINITION 6.1—SINGULAR VALUES
Given a matrix $X \in \mathbb{R}^{K \times J}$. The singular values of X are the nonnegative square roots of the eigenvalues of $X^T X$. □

DEFINITION 6.2—SINGULAR VALUE DECOMPOSITION
The singular value decomposition of a matrix $X \in \mathbb{R}^{K \times J}$ is the decomposition

$$X = U \Sigma V^T \tag{6.5}$$

where $U \in \mathbb{R}^{K \times K}$ is a unitary matrix such that $UU^T = I$, $V \in \mathbb{R}^{J \times J}$ is a unitary matrix such that $VV^T = I$, and $\Sigma \in \mathbb{R}^{K \times J}$ is a diagonal matrix with the singular values of X, σ_i, on the diagonal, ordered from the largest to the smallest. □

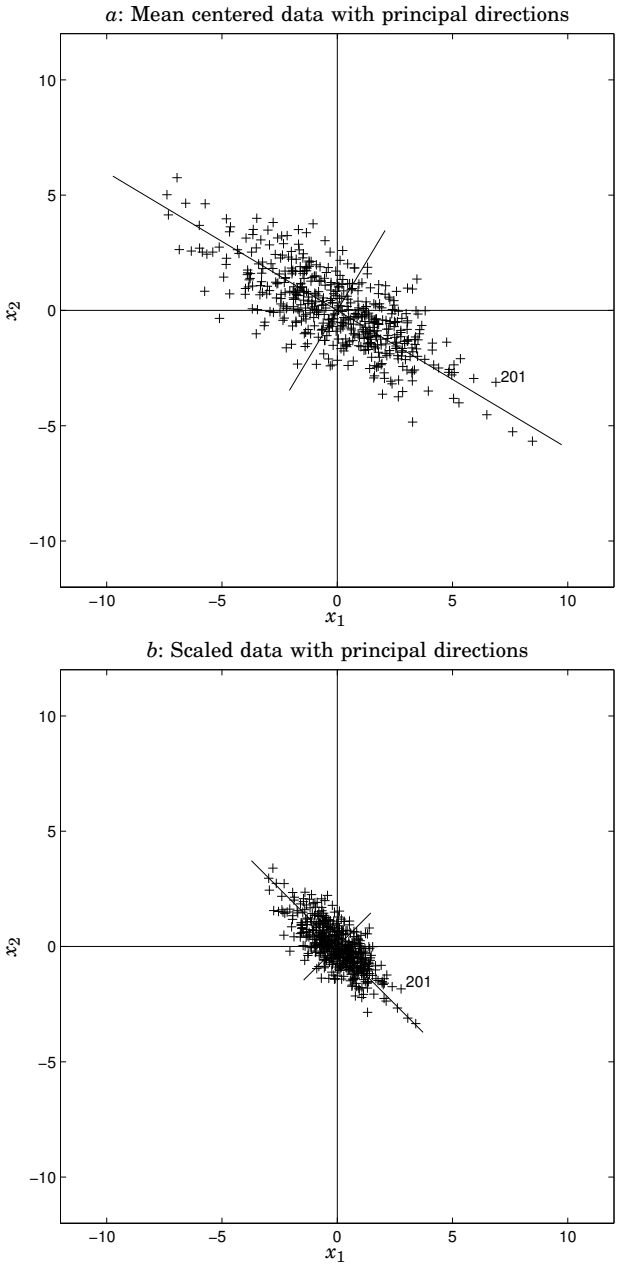

Figure 6.2 PCA performed after mean centering (a) and after mean centering and scaling to unit variance in the original variables (b). The directions of the principal components are given by the orthogonal lines, having the length of 4σ, i.e., standard deviations.

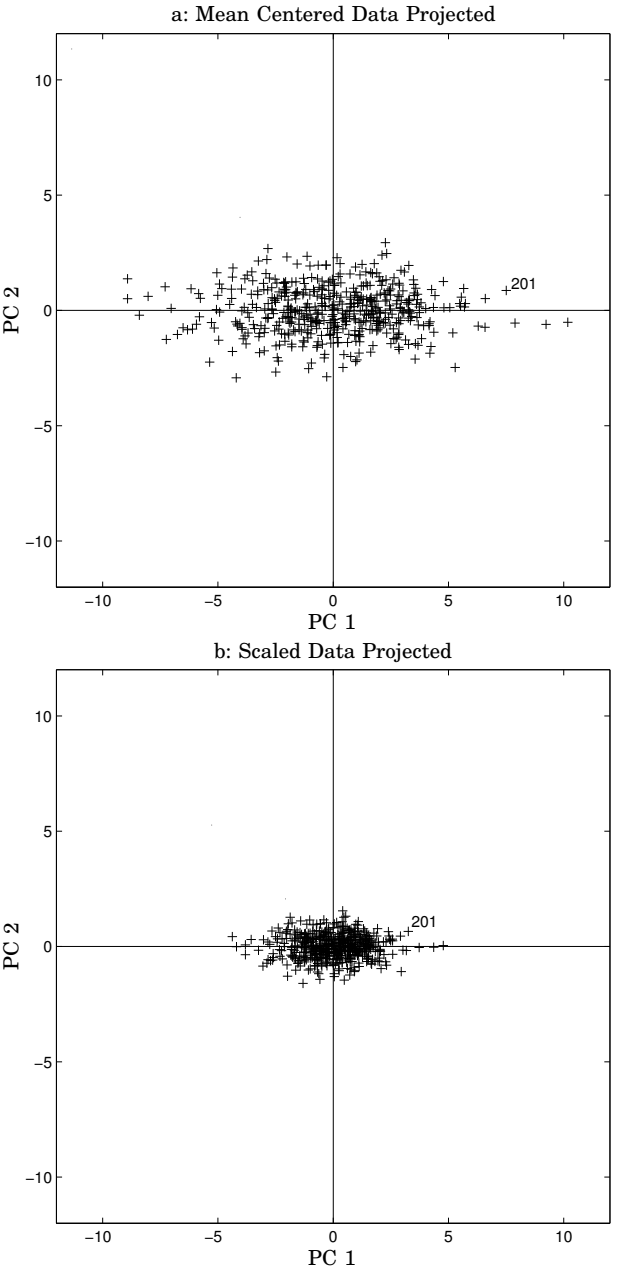

Figure 6.3 Projection of the original variables on to the space spanned by the principal components after mean centering (a) and after mean centering and scaling to unit variance in the original variables (b).

The right eigenvector associated with the largest singular value, v_1 in V, has the same direction as the first principal component, p_1. The right eigenvector associated with the second largest singular value, v_2 in V, determines the direction of the second principal component, p_2, and so on. Thus, the loadings are the first R right eigenvectors V_R and the scores are $T_R = U_R \Sigma_R$. The maximum number of eigenvectors equals the number of rows (or columns) of X.

If SVD is performed on X^T and X in the covariance matrix $X^T X$ the decomposition becomes

$$X^T X = V \Sigma U^T U \Sigma V^T = V \Sigma^T \Sigma V^T \Longrightarrow X^T X V = V \Sigma^T \Sigma \tag{6.6}$$

which is the eigenvalue decomposition of $X^T X$, since Σ is diagonal, and the eigenvectors, i.e., the principal components, are the columns of V.

The selection of the number of principal components to keep in the model is based on some criterion, e.g., the size of the singular values in Σ or by cross validation. Only the first R eigenvectors, V_R, are kept in the model. The scores, T_R, are then calculated according to

$$T_R = X V_R \tag{6.7}$$

and the model becomes

$$X = T_R V_R^T + E_R \tag{6.8}$$

Note that the computation of $X^T X$ may result in reduced numerical precision, and it is therefore not recommended. Instead SVD should be performed on X directly.

If the singular values of X are plotted one gets a diagram where the singular values become smaller and smaller. If there is a sharp fall off after a certain number this may indicate that only the principal components up to this number are relevant and the last ones contain only the noise of the data. The variance accounted for, VAF, described by the rth principal component can then be described by

$$VAF_r = \frac{\sigma_r^2}{\Sigma^T \Sigma} \cdot 100\% \tag{6.9}$$

where σ_r is the rth singular value in Σ. This can also be used as a criterion for how many principal components to keep in the model.

Cross validation is performed by dividing the data X into a number of subsets and then build parallel models leaving out one of the subsets at a time. The sum of squares for the difference between the data X and the calculated data, i.e., $E_R = X - T_R V_R^T$ for all the models is collected to form the predictive sum of squares (PRESS). When adding another principal component does not give any significant reduction of the PRESS no more principal components are added to the model.

Nonlinear Iterative Partial Least Squares Nonlinear iterative partial least squares or NIPALS [Wold, 1966; Wold, 1975] is as the name indicates an iterative procedure used to calculate the loadings p and the scores t in PCA. The NIPALS algorithm is as follows

1. Choose a starting t
2. $p = X^T t/(t^T t)$
3. $p = p/ \parallel p \parallel$
4. $t = Xp/(p^T p)$
5. Back to 2 if t has not converged
6. $X = X - tp^T$
7. Back to 1

In the inner iteration, steps 2–5, the direction describing the largest variation in X is calculated. Then in step 6 this direction is subtracted from X and the search for the next direction is started.

NIPALS is efficient for large matrices and can easily handle missing data, see, e.g., [Nelson *et al.*, 1996]. However, it is not as robust as SVD and if the ratio between two successive eigenvalues is close to unity the rate of convergence is very slow. Therefore it is stated in the literature that NIPALS should not be used unless there is missing data in the matrix X. The NIPALS algorithm is closely related to the power method described in [Golub and Van Loan, 1996].

Monitoring of Continuous Processes using PCA

Monitoring of processes using statistical methods is also called statistical process control (SPC). The goal of SPC is to allow normal variations of the measured variables of a process within certain control limits, e.g., the temperature may vary between T_{high} and T_{low} during drying of some material, assuming this will lead to the specified product quality. The goal is also to detect any anomalous variations as fast as possible without any false alarms. SPC uses probability distributions, e.g. the normal distribution, to explain the normal behavior of the process and to calculate the control limits, i.e., the alarm thresholds of the variables, e.g, three standard deviations or 99% limits.

Different charts for supervision of the process can be used such as Shewhart control charts, cumulative sum, CUSUM, charts, and exponentially moving average, EWMA, charts, see, e.g., [Oakland, 1999]. These methods are all univariate methods, i.e., they only consider one variable at a time. When there is an interaction/cross-correlation between the variables the univariate SPC methods might fail to detect abnormal situations.

Information about such interactions can be extracted from the covariance matrix $X^T X$. Methods for doing this is known under the name multivariate SPC (MSPC) [Wise and Ricker, 1989; MacGregor and Kourti, 1995]. In MSPC the most commonly used method to extract independent components is PCA as described above. Then each observation in time $x(k) \in \mathbb{R}^J$ of the J variables is projected on to the score space $t_R(k) \in \mathbb{R}^R$ by multiplying $x(k)$ with the loading vectors of the model.

$$t_R(k) = P_R^T x(k) \tag{6.10}$$

The PCA model also gives an estimate for $x(k)$ by projecting the scores back to the original space according to

$$\hat{x}(k) = P_R t_R(k) = P_R P_R^T x(k) \tag{6.11}$$

giving the residual

$$\tilde{x}(k) = x(k) - \hat{x}(k) \tag{6.12}$$

where $\hat{x}$ is orthogonal to $\tilde{x}$. The PCA model divides the measurements from the process into two orthogonal subspaces, i.e., the subspace spanned by the R first principal components with the value $\hat{x}(k)$ and the subspace of the residuals with the value $\tilde{x}(k)$. Monitoring of the process takes place in both these subspaces using the Hotelling T^2 statistics and squared prediction error SPE as indices.

The T^2 statistics is used to monitor the scores in the principal component subspace. T^2 is a measure of the distance of $t_R(k)$ from the origin and it is defined as

$$T^2(k) = t_R(k)^T S_R^{-1} t_R(k) = x(k)^T P_R S_R^{-1} P_R^T x(k) \tag{6.13}$$

where $S_R \in \mathbb{R}^{R \times R}$ is a matrix with the R first eigenvalues of the mean centered $X^T X$ on the diagonal or in other words $S_R = \Sigma_R^T \Sigma_R$ from Eq. 6.5 after mean centering. If the vector $t_R(k) \in \mathbb{R}^R$ of a new observation is assumed to be Gaussian multivariate-distributed with zero mean and unit covariance matrix $N_R(0, \frac{S_R}{K-1})$ then T^2 is F-distributed [Johnson and Wichern, 1998] according to

$$T^2 \sim \frac{(K+1)R}{K(K-R)} F_{R,K-R}(\alpha) \tag{6.14}$$

where $F_{R,K-R}(\alpha)$ is the upper $100(1-\alpha)$ percentile of the $F_{R,K-R}$ distribution with R and $K-R$ degrees of freedom. R is the number of principal components in the model and K is the number of samples used to build the PCA model.

The residual subspace could in principle be monitored by a T^2 statistics using the last $J-R$ principal components not used in the model [Johnson and Wichern, 1998], but if there exists linear dependencies in the data and the rank of X is not full or if the least significant singular values are very small the matrix to be inverted will be close to singular and the solution will be very numerically sensitive. Instead the residual space is monitored using the squared prediction error or SPE

$$SPE(k) = x(k)^T(I - PP^T)x(k) \tag{6.15}$$

which is the part of $x(k)$ not modeled by the R principal components squared. If $x(k)$ is an observation from a multivariate normal distribution, $N_J(0, \hat{\Sigma})$, the control limits for the SPE can be approximated by [Nomikos and MacGregor, 1995b]

$$SPE_{lim}(\alpha) = g\chi^2_h(\alpha) \tag{6.16}$$

where $g = v/2m$ and $h = 2m^2/v$ with m and v being the sample mean and sample variance of the SPE for the data used in the model. $\chi^2_h(\alpha)$ is the upper $100(1-\alpha)$ percentile of the χ^2_h distribution with h degrees of freedom. Another approximation of the χ^2 distribution of the SPE can be found in [Jackson and Mudholkar, 1979].

Monitoring using a combination of T^2 and SPE has been suggested in the literature [Yue and Qin, 2001]. The individual scores can also be monitored by using bivariate plots of the scores from two principal components at a time and calculate control limits on the form of ellipses.

When the process is detected to be outside its control limits in the scores, T^2, or SPE the contribution from the original variables to the deviations can be calculated to determine the cause [Miller *et al.*, 1998]. Contribution plots are described below for batch process monitoring.

The steps for the monitoring of a process using MSPC are

- Gather sufficient amount of data from the process running under normal conditions and store it in the data matrix X.
- Preprocess the data to find outliers, center, and scale the data.
- Calculate the loading vectors for X and determine how many principal components to keep in the model.
- Calculate the control limits.
- For a new sample calculate the scores, T^2 and, SPE. It is important to use the same centering and scaling as for the model data.
- If the process is outside the control limits try to determine the cause, e.g., by using contribution plots.

Dynamic Principal Component Analysis

Applying PCA on a data matrix X gives a linear static model even though the data might contain dynamic information, since PCA assumes there is no correlation over time. When this is the case the exact relations between the variables will not be revealed. The scores will be auto-correlated and may also be cross-correlated and the statistical basis is lost. To handle this dynamic PCA or DPCA for continuous processes was developed in [Ku *et al.*, 1995]. DPCA is ordinary PCA performed on the matrix $X_d \in \mathbb{R}^{(K-d)\times J(d+1)}$ constructed from $X \in \mathbb{R}^{K\times J}$ by time shifting the data d samples according to

$$X_d = \begin{bmatrix} x(1)^T & x(2)^T & \dots & x(d+1)^T \\ x(2)^T & x(3)^T & \dots & x(d+2)^T \\ \vdots & \vdots & \ddots & \vdots \\ x(K-d)^T & x(K-d+1)^T & \dots & x(K)^T \end{bmatrix} \tag{6.17}$$

where $x(k)^T = [\, x_1(k)\ x_2(k)\ \dots\ x_J(k)]$ is the measurement vector of the J variables at time k. The idea is that the residuals from a model developed using X_d instead of X should be less correlated and would provide a better statistical basis. A method for determining the time lag d is proposed in [Ku *et al.*, 1995], which is based on singular values and auto- and cross-correlation plots of the scores.

A criteria for determining the number of principal components, R, and the time lag, d, is also given in [Ku *et al.*, 1995] as a design procedure

1. Start with $d = 0$
2. Construct X_d
3. Perform PCA
4. Set $j = J(d+1)$ and $r(d) = 0$
5. Does the jth principal component representing a linear relation? If not go to 7
6. $j = j - 1$ and $r(d) = r(d) + 1$. Go to 5
7. $r_{new}(d) = r(d) - \sum_{l=0}^{d-1}(d-i+1)r_{new}(i)$
8. If $r_{new}(d) \leq 0$ Stop
9. $d = d + 1$ go to 2

The steps 4–6 are the normal decision of the number of principal components to keep in the model, e.g., test if the jth singular value is insignificant. $r(l)$ is then the number of insignificant singular values, i.e., the number of principal components kept out of the model.

The design procedure is based on that the static relations will be determined when applying PCA on X_0 and that these will be repeated every time the matrix is extended and so will the dynamic relations. It is also important to examine the auto- and cross-correlation plots to determine if the remaining scores are independent. If they are not this indicates that more principal components should be added to the model to get better statistical properties for the residuals and hence the SPE.

As pointed out in [Ku *et al.*, 1995] the idea of using time shifted data for modeling dynamic systems is not new and has been used extensively in system identification, see, e.g., [Ljung, 1999] and that DPCA does not identify accurate time series models. The purpose of the method is to develop multivariate models for monitoring and the advantage is that it is a very simple method. Monitoring of continuous processes with DPCA is used in a similar way as PCA, described above, using T^2 and squared prediction error SPE.

In [Negiz and Cinar, 1997] it is shown that process noise greatly affects DPCA and a method based on canonical variate analysis is proposed instead. Also in [Kruger *et al.*, 2004] problems with DPCA are discussed. It is shown that if the process variables are uncorrelated and a DPCA model is built it will result in auto-correlated score variables. The auto-correlation will also increase if the process variables are highly correlated. This may result in an increase of false alarms. In [Kruger *et al.*, 2004] the use of auto-regressive moving average, ARMA, filters is introduced to remove the auto-correlation. In [Li and Qin, 2001] an indirect dynamic PCA method based on subspace model identification is proposed, which is able to give consistent models in the presence of measurement noise.

6.3 Multi-way Methods for Batch Process Monitoring Based on PCA

The finite duration and non-linear behavior of batch processes where the variables change significantly over time means that monitoring of batch processes is considerably different from monitoring of continuous processes. Therefore different methods have been developed in the multivariate statistical field to take the differences from continuous processes into account.

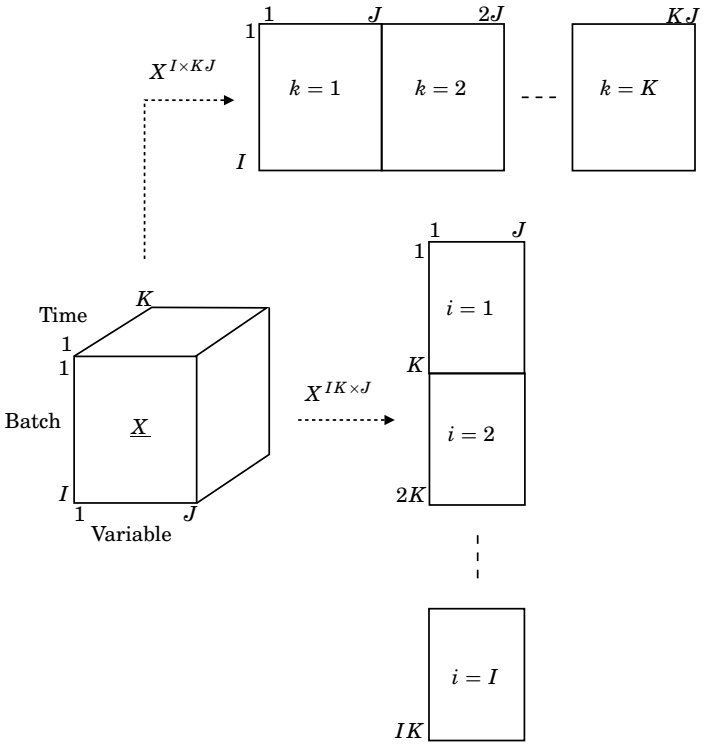

Figure 6.4 The third-order tensor $\underline{X}$ can be unfolded into several two dimensional matrices. $X^{I\times KJ}$ and $X^{IK\times J}$ are two of these ways.

Multi-way Principal Component Analysis

Multi-way principal component analysis [Wold *et al.*, 1987], MPCA (also called Tucker1), is a method for performing PCA on third-order tensors. The name is ambiguous since multi-way principal component analysis also means the extension of PCA to more than two dimension, therefore MPCA is also called unfolded PCA. The name MPCA will be used here.

Measurements collected from a batch process result in a third-order tensor of data $\underline{X} \in \mathbb{R}^{I\times J\times K}$, see also Fig. 6.4. In MPCA $\underline{X}$ is unfolded into a matrix, X, and then ordinary PCA is performed on the unfolded data. The tensor $\underline{X}$ can be unfolded in different ways to describe different modes, or dimensions, of the data [Westerhuis *et al.*, 1999], see Fig. 6.4.

In batch process monitoring the variation between batches is of most interest and the unfolding of $\underline{X}$, which describes this best is when each of its vertical slices $(I \times J)$ are placed side by side in the two-dimensional

batch-wise unfolded matrix $X^{I\times KJ} \in \mathbb{R}^{I\times KJ}$ [Westerhuis *et al.*, 1999; Kourti, 2003b]

$$X^{I\times KJ} = \begin{bmatrix} x^1(1)^T & x^1(2)^T & \dots & x^1(k)^T & \dots & x^1(K)^T \\ x^2(1)^T & x^2(2)^T & \dots & x^2(k)^T & \dots & x^2(K)^T \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ x^i(1)^T & x^i(2)^T & \dots & x^i(k)^T & \dots & x^i(K)^T \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ x^I(1)^T & x^I(2)^T & \dots & x^I(k)^T & \dots & x^I(K)^T \end{bmatrix} \tag{6.18}$$

where

$$x^i(k)^T = \begin{bmatrix} x_1^i(k) & x_2^i(k) & \dots & x_J^i(k) \end{bmatrix} \tag{6.19}$$

is a vector containing the measurements of the J variables at time k for batch i. By subtracting the mean from each column in $X^{I\times KJ}$ the average trajectory of each variable over time is removed and MPCA will model the variation around these trajectories. The loadings calculated from this unfolding have entries corresponding to each variable at each time with the length of the loading vector being KJ.

MPCA on the unfolding $X^{I\times KJ}$ was first used for batch process monitoring by Nomikos and MacGregor in [Nomikos and MacGregor, 1994; Nomikos and MacGregor, 1995b; Nomikos and MacGregor, 1995a]. One drawback with the method is that it assumes that data for the complete batch is available. If it is to be used for on-line monitoring future measurements must be estimated. In [Nomikos and MacGregor, 1995b] three different methods for anticipating the future measurements in a new monitored batch are suggested. The first sets all future measurements to zero, which is equal to assuming that the batch will follow the nominal trajectory if the data is mean centered. The second assumes that the current deviation from the nominal trajectory will remain the same to the end of the batch, i.e., it fills in the rest of the trajectories with the current values of the variables. The third uses the PCA model's ability to predict the future deviations from the nominal trajectory by projection to the model plane. The unknown future deviations are regarded as missing values and the principal components of the model built from nominal batches are used to predict these under the restriction that the already measured values up to time k and the correlation structure described by the loadings are kept the same. The scores at time k, $t(k) \in \mathbb{R}^R$, of the running batch are calculated by least squares estimation

$$t(k) = \left(P(k)^T P(k)\right)^{-1} P(k)^T x_{new}(k) \tag{6.20}$$

where $P(k) \in \mathbb{R}^{kJ \times R}$ consists of the first kJ rows of the loading matrix $P \in \mathbb{R}^{KJ \times R}$ from the model and $x_{new}(k) \in \mathbb{R}^{kJ \times 1}$ are the measured variables of the running batch up to time k.

Several methods for prediction of future measurements have been proposed in the literature. In [Arteaga and Ferrer, 2002] different methods are described and compared, and it is shown that several of them are equivalent. In [García-Muñoz *et al.*, 2003] also several methods are compared for batch process monitoring. One drawback of having to predict future values is that the prediction does not become reliable until a certain amount of the measurements for a batch have been collected. For example, it has been reported in the literature that approximately 10% of the batch needs to be measured for the projection to model plane method. At the end of the batch the entire history of the batch has been taken into account calculating the score of the batch.

The unfolding $X^{IK \times J}$ in Fig. 6.4 was used for batch process monitoring in [Wold *et al.*, 1998] to overcome the problem with the need to have the measurements for the full batch.

$$X^{IK \times J} = \begin{bmatrix} x^1(1)^T \\ x^1(2)^T \\ \vdots \\ x^1(K)^T \\ \vdots \\ x^I(1)^T \\ x^I(2)^T \\ \vdots \\ x^I(K) \end{bmatrix} \tag{6.21}$$

This unfolding preserves the direction of the variables. When the *variable-wise unfolded* matrix $X^{IK \times J}$ is mean centered over its columns the grand mean of each variable over all batches and all times is subtracted from the trajectory of each variable in each batch. This leaves the time-varying part of the trajectory in the data that PCA tries to model. This unfolding also leads to that there will be J entries in the loading for each principal component, which will remain constant over the duration of the whole batch. It implies that the correlation among the variables is the same during the whole batch, which necessarily does not have to be true. It may change several times during the batch. An alternative way is to remove the time-varying part of the trajectory by first mean centering and scaling the variables over the batch mode, i.e., in the same way as described above

for the batch-wise unfolded $X^{I\times KJ}$, and then unfold the tensor to $X^{IK\times J}$. The problem that the directions of the loadings are constant will still remain though. These directions will only be the true principal directions at certain times during the batch. This subject will be further discussed in connection to the moving window PCA method below.

A comparison of MPCA on the two different unfoldings is performed in [Westerhuis *et al.*, 1999] showing that unfolding $X^{IK\times J}$ usually needs almost as many principal components as there are original variables. Another discussion of the two different ways of unfolding, in [Kourti, 2003b], also states that the unfolding $X^{I\times KJ}$ is of most relevance when monitoring batch processes. The two approaches can also be combined where $X^{IK\times J}$ is used on-line during the batch and the unfolding $X^{I\times KJ}$ is used when the batch is finished to avoid being forced to fill in future measurements. In [Kourti, 2003a] another drawback with the unfolding $X^{IK\times J}$ is discussed, namely that the method is not capable of handling the case when a variable is only measured during a certain part of the batch duration.

In [Wold *et al.*, 1998] a slightly different approach is also taken where partial least squares, PLS, regression, see e.g., [Wold *et al.*, 1987; Wold *et al.*, 2001], is performed on $X^{IK\times J}$ and y, where y contains a measure of the maturity of the batch, e.g., the batch duration. This method is not treated here since the focus is on PCA based methods. Comments on the method in [Wold *et al.*, 1998] can be found in [Kourti, 2003a].

Control Limits To develop control limits for the T^2 and the SPE when using the unfolding $X^{I\times KJ}$ the normal batches are run through the chosen method for estimating future measurements as if they are new batches. The scores and SPE calculated for each batch at each time interval are used to calculate the T^2 and the confidence limits under the assumption that they are normally distributed. T^2 is calculated in the same way as in Eq. 6.13

$$T^2(k) = t(k)^T \hat{S}^{-1}(k) t(k) \tag{6.22}$$

but now $t(k)$ are the predicted scores from Eq. 6.20 and $\hat{S}(k)$ is an estimate calculated using the predicted scores from all the batches filled from time k.

Since PCA is used to calculate the model the control limits will be similar to the ones calculated for a continuous process above. The difference is that the degrees of freedom will change. T^2 will now be distributed according to

$$T^2 \sim \frac{(I^2-1)R}{I(I-R)} F_{R,I-R}(\alpha) \tag{6.23}$$

SPE in MPCA is only calculated for the last sample at time instance

k according to

$$SPE(k) = \sum_{l=(k-1)J+1}^{kJ} e_k(l)^2 \tag{6.24}$$

where e_k is the residuals given by

$$e_k = x_{new}(k) - t(k)P^T \tag{6.25}$$

This gives a measure of the orthogonal distance of the batch to the model described by the R principal components at time instance k. The control limit for the SPE is calculated using Eq. 6.16 with the mean m and variance v calculated from the normal batches. The control limits for the SPE will change with time since the sample mean m and sample variance v from the normal batches change with k.

When using the unfolding $X^{IK\times J}$ the control limits for the scores are calculated from the standard deviation of scores from the batches used to build the model [Wold *et al.*, 1998]. The calculated values for each score t_r over time for each batch is reshaped to the matrix X_{t_r} according to

$$X_{t_r} = \begin{bmatrix} t_r^1(1) & \dots & t_r^1(k) & \dots & t_r^1(K) \\ \vdots & \ddots & \vdots & & \vdots \\ t_r^i(1) & \dots & t_r^i(k) & \dots & t_r^i(K) \\ \vdots & & \vdots & \ddots & \vdots \\ t_r^I(1) & \dots & t_r^I(k) & \dots & t_r^I(K) \end{bmatrix} \tag{6.26}$$

where $t_r^i(k)$ is the value of the rth score of batch i at time k. The mean and standard deviation from these matrices are used for the control limits when monitoring new batches. Both the mean and the standard deviation are changing over time. T^2 and SPE can be used for this unfolding as well.

Contribution Plots In case of a fault one of the T^2 or SPE statistics, or both, will hopefully detect it and indicate this by going outside the limits. When a fault is detected the model can be used to determine which of the original J variables contributed the most by using contribution plots [Nomikos, 1996; Westerhuis *et al.*, 2000] to determine what type of fault has occurred. The T^2 and SPE plots only indicate that the process is not operating according to normal operation and do not give information about what the root cause is.

If only the T^2 goes out of its limits the model is still valid, but the distance from the monitored batch to the center of the model is larger

than normal. In this case, the contribution of each variable to the T^2 should be examined. If instead SPE is higher than its control limit this indicates that the model is no longer valid and that this behavior was not present in the batches used for calculating the model. In this case, the contribution of each variable to the SPE should be examined.

One way of calculating the contribution to T^2 is to calculate the jth variable's contribution at each time instance k using [Nomikos, 1996; Westerhuis *et al.*, 2000]

$$c_{kj}^{T^2} = \sum_{r=1}^{R} \sigma_r^{-1} t_r x_j(k) P_{kj,r} \tag{6.27}$$

where σ_r is the rth diagonal element in S in Eq. 6.13, t_r is the rth score for the monitored batch, $x_j(k)$ is the measurement of the jth variable at time k, and $P_{kj,r}$ is the kjth element in the rth loading vector. Eq. 6.27 assumes that the measurements for the whole batch are available to calculate the rth score t_r for the finished batch. If future measurements are approximated using, e.g., projection to model plane, these will also influence the contribution plots and the contribution plots are therefore not fully reliable.

Another way is to calculate contributions to scores that are out of their individual limits [Miller *et al.*, 1998]. For each score t_r, which is outside its limits the contribution from variable j is calculated by

$$c_{kj}^{t_r} = x_j(k) P_{kj,r} \tag{6.28}$$

Both approaches assume that the loadings in the model are orthonormal and the scores are orthogonal, which is the case when using PCA. Extensions to non-orthogonal cases can be found in [Westerhuis *et al.*, 2000].

The contribution of the jth variable at time k to the SPE can be calculated as

$$c_{kj}^{SPE} = (e_j(k))^2 = (x_j(k) - \hat{x}_j(k))^2 \tag{6.29}$$

where $x_j(k)$ is the measurement of variable j at time k from the monitored batch and $\hat{x}_j(k)$ is the part described by the PCA model.

When monitoring a batch online and filling in estimates of future measurements these will also give contributions to the scores, T^2, and SPE. Using the projection to model plane method the scores at time k, $t(k)$, are used to calculate the future values $x_{new,k}^{future} \in \mathbb{R}^{(K-k)J \times 1}$ from time $k+1$ to K according to

$$x_{new,k}^{future} = P^{future} t(k) \tag{6.30}$$

where $P^{future} \in \mathbb{R}^{(K-k)J \times R}$ are the loadings associated with the future predicted variables. The full, partially estimated, trajectories of the batch currently being monitored can now be written using the available measurement up to time k, $x_{new,k}$ according to

$$x^{full,k} = \left[x_{new,k} \; x_{new,k}^{future} \right] \tag{6.31}$$

This together with, e.g., Eq. 6.27 gives that the contribution to T^2 is

$$c_{\hat{k}j}^{T^2}(k) = \sum_{r=1}^{R} \hat{\sigma}_r^{-1}(k) t_r(k) x_j^{full,k}(\hat{k}) P_{\hat{k}j,r} \tag{6.32}$$

where $\hat{\sigma}_r(k)$ is the rth diagonal element in $\hat{S}(k)$ in Eq. 6.22 and $x_{\hat{k}j}^{full}(k)$ is the value of the jth variable, either measured (if $\hat{k} < k$) or estimated (if $\hat{k} > k$). Now two time indices are used, k which is the latest sample time, i.e., how long the batch has been running, and $\hat{k}$ which is the contribution time which can take a value between 1 and K. A discussion of this can also be found in [Meng *et al.*, 2003].

Another way of isolating faults in multivariate statistical methods is to use fault signatures in the scores and the residuals. This is described in [Yoon and MacGregor, 2001a].

Batch Dynamic PCA

The batch dynamic PCA method, BDPCA, was proposed in [Chen and Liu, 2002]. A correlation matrix based on time lagged data for one batch at the time is constructed and then an average of the matrices for all batches is used to build a PCA model. Here a new algorithm is used and it will be shown that instead of constructing several correlation matrices a model can be built directly from the SVD of one matrix constructed from time lagged data and that this method does not really result in a dynamic model.

In [Chen and Liu, 2002] the authors claim that MPCA using the unfolding $X^{I \times KJ}$ is not taking the dynamic relationship into account as a motivation for BDPCA. They base this claim on the calculation of the covariance matrix $S_{I \times I} \in \mathbb{R}^{I \times I}$ given by

$$S_{I \times I} = \frac{(X^{I \times KJ})(X^{I \times KJ})^T}{KJ - 1} \tag{6.33}$$

where

$$(X^{I\times KJ})(X^{I\times KJ})^T = \begin{bmatrix} \sum_{k=1}^{K}\sum_{j=1}^{J}(x_j^1(k))^2 & \sum_{k=1}^{K}\sum_{j=1}^{J}x_j^1(k)x_j^2(k) & \dots & \sum_{k=1}^{K}\sum_{j=1}^{J}x_j^1(k)x_j^I(k) \\ \sum_{k=1}^{K}\sum_{j=1}^{J}x_j^2(k)x_j^1(k) & \sum_{k=1}^{K}\sum_{j=1}^{J}(x_j^2(k))^2 & \dots & \sum_{k=1}^{K}\sum_{j=1}^{J}x_j^2(k)x_j^I(k) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{k=1}^{K}\sum_{j=1}^{J}x_j^I(k)x_j^1(k) & \sum_{k=1}^{K}\sum_{j=1}^{J}x_j^I(k)x_j^2(k) & \dots & \sum_{k=1}^{K}\sum_{j=1}^{J}(x_j^I(k))^2 \end{bmatrix} \tag{6.34}$$

and draw the conclusion that this matrix does not describe any relationship between the different time instances of the batches but only relationship between batches.

The problem is that this is the covariance matrix associated with the left singular vector of the SVD of $X^{I\times KJ}$ and not the right singular vector, which contain the loadings in the PCA model. If instead the correct covariance matrix $S_{KJ\times KJ} \in \mathbb{R}^{KJ\times KJ}$ is calculated

$$S_{KJ\times KJ} = \frac{(X^{I\times KJ})^T(X^{I\times KJ})}{I-1} \tag{6.35}$$

where

$$(X^{I\times KJ})^T(X^{I\times KJ}) = \begin{bmatrix} \tilde{S}_{1,1} & \tilde{S}_{1,2} & \dots & \tilde{S}_{1,k_2} & \dots & \tilde{S}_{1,K} \\ \tilde{S}_{2,1} & \tilde{S}_{2,2} & \dots & \tilde{S}_{2,k_2} & \dots & \tilde{S}_{2,K} \\ \vdots & \vdots & & \vdots & & \vdots \\ \tilde{S}_{k_1,1} & \tilde{S}_{k_1,2} & \dots & \tilde{S}_{k_1,k_2} & \dots & \tilde{S}_{k_1,K} \\ \vdots & \vdots & & \vdots & & \vdots \\ \tilde{S}_{K,1} & \tilde{S}_{K,2} & \dots & \tilde{S}_{K,k_2} & \dots & \tilde{S}_{K,K} \end{bmatrix} \tag{6.36}$$

with $\tilde{S}_{k_1,k_2} \in \mathbb{R}^{J\times J}$ equal to

$$\tilde{S}_{k_1,k_2} = \begin{bmatrix} \sum_{i=1}^{I} x_1^i(k_1)x_1^i(k_2) & \sum_{i=1}^{I} x_1^i(k_1)x_2^i(k_2) & \dots & \sum_{i=1}^{I} x_1^i(k_1)x_J^i(k_2) \\ \sum_{i=1}^{I} x_2^i(k_1)x_1^i(k_2) & \sum_{i=1}^{I} x_2^i(k_1)x_2^i(k_2) & \dots & \sum_{i=1}^{I} x_2^i(k_1)x_J^i(k_2) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{I} x_J^i(k_1)x_1^i(k_2) & \sum_{i=1}^{I} x_J^i(k_1)x_2^i(k_2) & \dots & \sum_{i=1}^{I} x_J^i(k_1)x_J^i(k_2) \end{bmatrix} \tag{6.37}$$

then it can be seen that this covariance matrix clearly is a measure of the relationship in time of the measurements and thus MPCA does take dynamics into account in the same way as in BDPCA. Below it will be shown that the two different unfolding MPCA methods described above are special cases of BDPCA.

The method proposed in [Chen and Liu, 2002] for calculating an BDPCA model is to order the measurements from batch i in the matrix $X_d^i \in \mathbb{R}^{(K-d)\times(d+1)J}$ according to

$$X_d^i = \begin{bmatrix} x^i(1)^T & x^i(2)^T & \dots & x^i(d+1)^T \\ x^i(2)^T & x^i(3)^T & \dots & x^i(d+2)^T \\ \vdots & \vdots & \ddots & \vdots \\ x^i(K-d)^T & x^i(K-d+1)^T & \dots & x^i(K)^T \end{bmatrix}, \tag{6.38}$$

to include dynamic effects in the same way as in DPCA, described above. The measurements from J variables of batch i at time k, $x^i(k)^T$, are defined by Eq. 6.19. A covariance matrix, $S^i_{X_dX_d} \in \mathbb{R}^{(d+1)J\times(d+1)J}$, is then calculated for each of the I batches according to

$$S^i_{X_d^iX_d^i} = \frac{\left(X_d^i\right)^T\left(X_d^i\right)}{K-d-1} \tag{6.39}$$

where

$$(X_d^i)^T(X_d^i) = \begin{bmatrix} \bar{S}^i{}_{1,1} & \bar{S}^i{}_{1,2} & \dots & \bar{S}^i{}_{1,k_2} & \dots & \bar{S}^i{}_{1,d+1} \\ \bar{S}^i{}_{2,1} & \bar{S}^i{}_{2,2} & \dots & \bar{S}^i{}_{2,k_2} & \dots & \bar{S}^i{}_{2,d+1} \\ \vdots & \vdots & & \vdots & & \vdots \\ \bar{S}^i{}_{k_1,1} & \bar{S}^i{}_{k_1,2} & \dots & \bar{S}^i{}_{k_1,k_2} & \dots & \bar{S}^i{}_{k_1,d+1} \\ \vdots & \vdots & & \vdots & & \vdots \\ \bar{S}^i{}_{d+1,1} & \bar{S}^i{}_{d+1,2} & \dots & \bar{S}^i{}_{d+1,k_2} & \dots & \bar{S}^i{}_{d+1,d+1} \end{bmatrix} \tag{6.40}$$

with $\bar{S}^i{}_{k_1,k_2} \in \mathbb{R}^{J \times J}$ equal to

$$\bar{S}^i{}_{k_1,k_2} = \begin{bmatrix} \sum_{k=0}^{K-d-1} x_1^i(k+k_1)x_1^i(k+k_2) & \dots & \sum_{k=0}^{K-d-1} x_1^i(k+k_1)x_J^i(k+k_2) \\ \vdots & \ddots & \vdots \\ \sum_{k=0}^{K-d-1} x_J^i(k+k_1)x_1^i(k+k_2) & \dots & \sum_{k=0}^{K-d-1} x_J^i(k+k_1)x_J^i(k+k_2) \end{bmatrix}. \tag{6.41}$$

Then an average covariance matrix $S^{avg}_{X_dX_d}$ using the I covariance matrices is calculated.

$$S^{avg}_{X_dX_d} = \frac{(K-d-1)\sum_{i=1}^{I} S^i_{X_dX_d}}{I(K-d)-1} \tag{6.42}$$

Finally PCA is applied to the average covariance matrix $S^{avg}_{X_dX_d}$ to build the model.

To determining the time lag d in BDPCA it is proposed in [Chen and Liu, 2002] that the method for DPCA in [Ku *et al.*, 1995] should be used. This method is described above in the part describing DPCA in Section 6.2.

Alternative Algorithm for BDPCA To avoid having to calculate the covariance matrix for each batch and then average over all the batches a new algorithm is proposed. If the time lagged matrices for all I batches, X_d^i, are included after each other in the matrix $X_d \in \mathbb{R}^{I(K-d)\times J(d+1)}$ according to

$$X_d = \begin{bmatrix} X_d^1 \\ \vdots \\ X_d^I \end{bmatrix} \tag{6.43}$$

the covariance matrix, $S_{X_d X_d}$, for all I batches can be calculated according to

$$S = \frac{X_d^T X_d}{I(K-d)-1} = \frac{1}{I(K-d)-1}\left[(X_d^1)^T\ (X_d^2)^T\ \ldots\ (X_d^I)^T\right]\begin{bmatrix} X_d^1 \\ X_d^2 \\ \vdots \\ X_d^I \end{bmatrix} = \frac{\sum_{i=1}^{I}(X_d^i)^T X_d^i}{I(K-d)-1}, \tag{6.44}$$

which is the same covariance matrix as in Eq. 6.42 and thus the PCA model can be determined from the SVD of the matrix X_d directly.

Relation to MPCA If no time lag is introduced, i.e., $d = 0$, in Eq. 6.43 the result becomes $X_0 \in \mathbb{R}^{IK \times J}$, which is equivalent to unfolding $X^{IK \times J}$ in Fig. 6.4 and Eq. 6.21 used in MPCA. On the other hand if the time lag reaches its maximum, i.e. $d = K - 1$, the result is $X_{K-1} \in \mathbb{R}^{I \times KJ}$, which is equivalent to unfolding $X^{I \times KJ}$ in Fig. 6.4 and Eq. 6.18 used in MPCA.

Mean Centering and Scaling A question not discussed in [Chen and Liu, 2002] is how to mean center and scale the matrix X_d. The mean of the columns and the size of the matrix X_d changes with the time lag d. The two extremes when $d = 0$ and $d = K - 1$ are considered in the section above describing MPCA. After correspondence with one of the authors [Chen, 2004] it was clarified that in [Chen and Liu, 2002] the batches are first unfolded to $X^{I \times KJ}$, mean centered and scaled to unit variance over its columns, and then refolded to the dynamic structure propose in the method. This removes the time-varying part of the trajectory in the same way as described above for the unfolding $X^{I \times KJ}$.

Shortcomings of BDPCA Looking at the new algorithm it becomes clear that the generated model only models the most dominant direction of the time lagged data over the batches since it mixes the time and batch directions. It does not describe the changing dynamics. This is the same problem as in MPCA using the unfolding $X^{IK \times J}$ where the loadings remain constant over the batch duration. BDPCA does not take the whole batch into account when calculating the scores for a new batch, only the last $d + 1$ samples are used. This also leads to that the control limits for BDPCA have to be calculated in a similar fashion as in variable-wise unfolded MPCA. Since it here has been proven that the BDPCA method is

quite different from the DPCA method it can be questioned if the method for determining the time lag in [Ku *et al.*, 1995] is a good choice. This is however not further investigated here.

Alternative MPCA Approaches

There are two major drawbacks with the presented methods. The first is the need to have all the measurements of the batch when using the unfolding $X^{I\times KJ}$ and the second that the unfolding $X^{IK\times J}$ and BDPCA are using only the most dominant direction over the batch when determining the model. Here are two alternative methods described that avoid these drawbacks.

Multi Model MPCA One easy way of eliminating the problem with having to have all the measurements of a batch is to build one model at each time instance k using the same approach as when using the unfolding $X^{I\times KJ}$. This will generate K different models built on the unfolded matrices $X^{I\times kJ}$, where $k = 1, 2, \ldots, K$. This approach therefore will here be called multi model MPCA and it uses the same theory as the batch-wise unfolded MPCA. The model includes one more measurement of all the variables for each time instance. The problem with this method is that a large number of models have to be both developed and also stored to be used during the monitoring of a new batch. It could be argued that it is too tedious having to determine the number of principal components to use in the model at each time instance and the storage requirements will be to large. But since only the singular values of the model batches need to be stored at the first stage of modeling this becomes manageable. Using plots of how the singular values change over time gives insight on how the dynamics of the batch are changing. After choosing the number of principal components to be used for the specific model at time k only the loadings for the model with the dimensions $R \times kJ$ need to be stored. A more serious problem is that the principal components will both change direction and significance, i.e. the direction of the first principal component will become less important and it will switch place with the second principal component. This is further discussed in Chap. 9.

If the method results in too many models a model can be calculated at, e.g., only every tenth sample together with filling in the future measurement with projection to the model plane between two models just as in the batch-wise unfolded MPCA.

Moving Window PCA Another method, which produces a model specific for each time instance is moving window PCA [Lopes and Menezes, 1998; Lennox *et al.*, 2001a; Lennox *et al.*, 2001b; Lennox *et al.*, 2002]. Instead of as in BDPCA constructing the matrix X_d^i according to Eq. 6.38

a model is calculated for each point in time k by constructing $X_d(k) \in \mathbb{R}^{I\times(d+1)J}$ according to

$$X_d(k) = \begin{bmatrix} x^1(k-d)^T & x^1(k-d+1)^T & \dots & x^1(k)^T \\ x^2(k-d)^T & x^2(k-d+1)^T & \dots & x^2(k)^T \\ \vdots & \vdots & \ddots & \vdots \\ x^I(k-d)^T & x^I(k-d+1)^T & \dots & x^I(k)^T \end{bmatrix} \tag{6.45}$$

where $k = d+1,\dots,K$. This will give $K-d$ different models where each model describes the behavior at time instance k based on measurements from $d+1$ time instances. The models are calculated from a moving window thus the name moving window PCA. The length of the window d does not have to be constant over the whole batch. The method can also easily be used for batches with different run-lengths.

If d is chosen to be constant over the whole batch duration it can be noticed that BDPCA models the most dominant direction of the $K-d$ models from moving window PCA. This can be seen if the matrix X_d in Eq. 6.43 is rearranged, by changing the rows, to

$$X_d = \begin{bmatrix} X_d(d+1) \\ \vdots \\ X_d(K) \end{bmatrix} \tag{6.46}$$

When $d = 0$ this gives K models where each model gives the relations of the variables at time point k. $X_0(k) \in \mathbb{R}^{I\times J}$ is given by

$$X_0(k) = \begin{bmatrix} x^1(k)^T \\ x^2(k)^T \\ \vdots \\ x^I(k)^T \end{bmatrix} \tag{6.47}$$

If the K matrices $X_0(k)$ for $k = 1,2,\dots,K$ are stacked on top of each other and the rows are reordered this will give the matrix $X^{IK\times J}$, and again it can be seen that this unfolding is an average model over all time instances.

If $d = K-1$ only one model can be calculated, $X_{K-1}(k) \in \mathbb{R}^{I\times KJ}$, given by

$$X_{K-1}(K) = \begin{bmatrix} x^1(1)^T & x^1(2)^T & \dots & x^1(K)^T \\ x^2(1)^T & x^2(2)^T & \dots & x^2(K)^T \\ \vdots & \vdots & \ddots & \vdots \\ x^I(1)^T & x^I(2)^T & \dots & x^I(K)^T \end{bmatrix} \tag{6.48}$$

where the only possible value for k is K. This equals MPCA using the unfolding $X^{I \times KJ}$ as described above.

There is no general method for determining the value of d in the method, e.g., in [Lennox *et al.*, 2002] it is stated that the length of the window has limited impact.

Other Related Methods

Other approaches for batch process monitoring include multi-block and hierarchical PCA, see e.g., [Kourti *et al.*, 1995; Westerhuis *et al.*, 1998]. In multi-block PCA the data is, as the name implies, divided into blocks. Each block is individually modeled using PCA and then the scores from the different models are combined to a super block. This super block is then modeled with PCA to get super scores and super weights. The difference between multi-block and hierarchical PCA is if the super weight or the super score is normalized to have unit length.

In [Rännar *et al.*, 1998] an hierarchical PCA is used in an adaptive manner to monitor batch processes. The blocks used in the method are equivalent to $X_0(k) \in \mathbb{R}^{I \times J}$ in Eq. 6.47. In the algorithm a PCA model is first built at time $k = 1$. Then for the next time instance a hierarchical model is built using the scores from the first sample and from a model built on data from $k = 2$, thus calculating super scores and super weights. The data from the new time instance is weighted into the model. In the next time step the super score from the previous time is used together with the scores from the model built using data from the third time instance and so on. Related to this is the recursive PCA method [Li *et al.*, 2000] for adaptive process monitoring.

A method for monitoring of different product grades or recipes in batch processes is proposed in [Lane *et al.*, 2001]. The method is based on a multi-group model instead of using one model for each grade/recipe.

6.4 Three-way Methods for Batch Process Monitoring

Three-dimensional data appears in several different fields of science, e.g, in economy, psychology, and analytical chemistry. Thus, several methods have been developed in these fields. Since batch process data is three dimensional by nature some of these methods have been used for modeling of batch processes, see [Dahl *et al.*, 1999; Westerhuis *et al.*, 1999; Wise *et al.*, 1999; Meng *et al.*, 2003; Louwerse and Smilde, 2000; Smilde, 2001]. A brief description of some of these methods is given below.

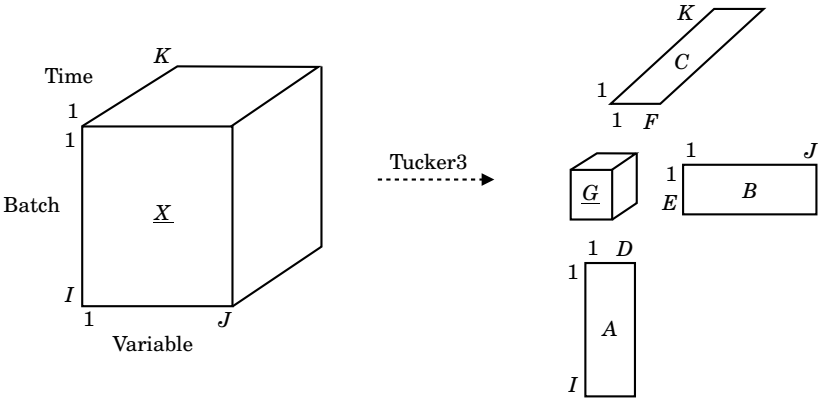

Figure 6.5 Tucker3 model of the third-order tensor $\underline{X}$ with the loadings A, B, and C, and the core tensor $\underline{G}$.

Tucker3

Several multi-way statistical methods come from the psychometrics (i.e. the psychological science of measuring cognitive or mental capacities, such as personality or intelligence) area where three-way or n-way data commonly exists. One of these methods is Tucker3 proposed by Tucker [Tucker, 1966], see also the book [Kroonenberg, 1983]. The structure of a Tucker3 model consists of a core cube, $\underline{G}$, and loadings (factors) A, B, and C for each data dimension, see Fig. 6.5.

If the third-order tensor $\underline{X} \in \mathbb{R}^{I \times J \times K}$ is unfolded to $X^{I \times KJ}$ then the model can be written as

$$X^{I \times KJ} = A G^{D \times EF} (C \otimes B)^T + \widetilde{E}^{I \times KJ} \tag{6.49}$$

where $\otimes$ is the Kronecker tensor product defined in Def. 6.3, $G^{D \times EF} \in \mathbb{R}^{D \times EF}$ is the unfolded core tensor $\underline{G} \in \mathbb{R}^{D \times E \times F}$, $\widetilde{E}$ is the residual matrix, and $A \in \mathbb{R}^{I \times D}$, $B \in \mathbb{R}^{K \times E}$, and $C \in \mathbb{R}^{J \times F}$ are the loadings of the three dimensions. The core cube $\underline{G}$ defines how the loading vectors interact with each other. This can be compared to the batch-wise unfolded MPCA and it can be seen that A can be seen as the scores and $G^{D \times EF} (C \otimes B)^T$ as the loadings in MPCA.

DEFINITION 6.3—KRONECKER TENSOR PRODUCT
For $H \in \mathbb{R}^{n \times m}$ the Kronecker tensor product is defined as

$$H \otimes L = \begin{bmatrix} h_{11}L & h_{12}L & \dots & h_{1m}L \\ h_{21}L & h_{22}L & \dots & h_{2m}L \\ \vdots & \vdots & \ddots & \vdots \\ h_{n1}L & h_{n2}L & \dots & h_{nm}L \end{bmatrix}$$

□

A Tucker3 model may have different numbers of components in each data direction, i.e, A, B, and C can have different number of columns. Tucker3 also has rotational freedom like PCA. If the core is constrained to only having ones on the super-diagonal the Tucker3 model becomes the PARAFAC model described below.

Algorithm for Tucker3 If B and C are given initial values and D, E, and F are the specified orders of the model, an iterative algorithm for Tucker3 is given by [Bro, 1998; Andersson and Bro, 1998]

1. $X^{I \times KJ}(C \otimes B) \overset{SVD}{=} U\Sigma V^T$
2. A is set to the D first columns of U
3. $X^{J \times KI}(C \otimes A) \overset{SVD}{=} U\Sigma V^T$
4. B is set to the E first columns of U
5. $X^{K \times JI}(B \otimes A) \overset{SVD}{=} U\Sigma V^T$
6. C is set to the F first columns of U
7. Go to 1 until convergence
8. $G = A^T X^{I \times KJ}(C \otimes B)$

One suggestion for the initialization of B and C is by computing $X^{J \times KI} \overset{SVD}{=} U\Sigma V^T$ and setting B to the E first columns of U. Then $X^{K \times JI} \overset{SVD}{=} U\Sigma V^T$ is computed and C is taken as the F first columns of U.

Monitoring using Tucker3 The scores for a new batch, a_{new}, are calculated by solving the least squares problem of Eq. 6.49 [Louwerse and Smilde, 2000] according to

$$a_{new} = (GG^T)^{-1} G \left[(C \otimes B)^T (C \otimes B)\right]^{-1} (C \otimes B)^T x_{new} \tag{6.50}$$

where $x_{new} \in \mathbb{R}^{KJ}$ are the measurements from the new batch. C; B, and $G = G^{D \times EF}$ from Eq. 6.49. The residual, e_{new} are calculated according to

$$e_{new} = x_{new} - (C \otimes B)G^T a_{new} \tag{6.51}$$

Then T^2 and SPE can be used in the same as for MPCA. For on-line monitoring a multi model approach can be adopted where a different Tucker3 model is built for each time instance.

Parallel Factor Analysis

Parallel factor analysis, or PARAFAC also called CANDECOMP [Harshman, 1970; Carroll and Chang, 1970], for three-way data has instead of one score vector and one loading vector as in PCA, one score vector and two loading vectors (sometimes all three are called loading vectors). The PARAFAC model does not require orthogonality to identify the model, as opposed to PCA. In most cases PARAFAC has a unique solution and the true relations will be found if the right number of components are used in the model [Bro, 1998].

A PARAFAC model of the third-order tensor $\underline{X} \in \mathbb{R}^{I \times J \times K}$ is given by three loadings matrices A, B, and C, see Fig. 6.6 according to

$$\underline{X}_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} + \underline{E}_{ijk} \tag{6.52}$$

where a_{ir}, b_{jr}, and c_{kr} are elements from $A \in \mathbb{R}^{I \times I \times R}$, $B \in \mathbb{R}^{I \times J \times R}$, and $C \in \mathbb{R}^{K \times I \times R}$ respectively and $\underline{X}_{ijk}$ is the ijkth element in $\underline{X}$ and E_{ijk} is the corresponding error element in $\underline{E}$.

If the Kronecker tensor product, see Def. 6.3, is used the model can also be written as

$$X^{I \times KJ} = \sum_{r=1}^{R} a_r \left(c_r^T \otimes b_r^T \right) + E^{I \times KJ} \tag{6.53}$$

where $X^{I \times KJ}$ is $\underline{X}$ unfolded according to Fig. 6.4 and $a_r \in \mathbb{R}^I$, $b_r \in \mathbb{R}^J$, and $c_r \in \mathbb{R}^K$ are the rth column vectors from A, B, and C respectively. This means that PARAFAC and MPCA gives the same solution up to a rotation.

The Khatri-Rao product [Rao and Mitra, 1971] is defined as

DEFINITION 6.4—KHATRI-RAO PRODUCT
The Khatri-Rao product of two matrices with the same number of columns, R, $B \in \mathbb{R}^{J \times R}$ and $C \in \mathbb{R}^{K \times R}$

$$\begin{aligned} C \circ B &= [\, c_1 \otimes b_1 \quad c_2 \otimes b_2 \quad \dots \quad c_R \otimes b_R \,] \\ &= [\, vec(c_1 b_1^T) \quad vec(c_2 b_2^T) \quad \dots \quad vec(c_R b_R^T) \,] \end{aligned}$$

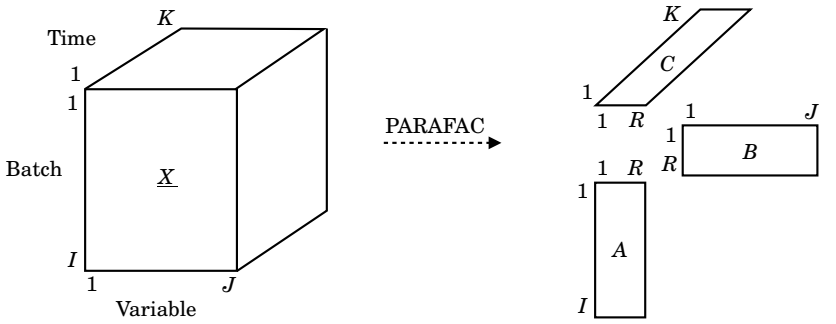

Figure 6.6 PARAFAC model of $\underline{X}$ consisting of the three loadings A, B, and C.

where c_r and b_r are the rth column from C and B respectively, and $vec(A)$ unfolds the matrix A column-wise to a column vector. □

Using the Khatri-Rao product the PARAFAC model can be written as [Bro, 1998]

$$X^{I\times KJ} = A\,(C \circ B)^T \tag{6.54}$$

As mentioned earlier PARAFAC can be seen as a constrained version of Tucker3, i.e., if the core in Tucker3 model only has ones at the super-diagonal it becomes a PARAFAC model. This implies that the number of components, R, must be the same in all directions and that a PARAFAC model uses fewer parameters than a Tucker3 model.

Algorithm for PARAFAC By using the notation described above the different ways of unfolding $\underline{X}$ can be written as

$$X^{I\times KJ} = A\,(C \circ B)^T \tag{6.55}$$

$$X^{J\times KI} = B\,(C \circ A)^T \tag{6.56}$$

$$X^{K\times JI} = C\,(B \circ A)^T \tag{6.57}$$

If B and C are given initial values an alternating least squares method [Bro, 1998] can be used by iterating the following three steps until A, B, and C converges.

1. $Z = (C \circ B)$, $A = X^{I\times KJ} Z \left(Z^T Z\right)^{-1}$
2. $Z = (C \circ A)$, $B = X^{J\times KI} Z \left(Z^T Z\right)^{-1}$
3. $Z = (B \circ A)$, $C = X^{K\times JI} Z \left(Z^T Z\right)^{-1}$

The solution after convergence may be a local minimum and several methods have been proposed to overcome this. A simple solution is, [Harshman and Lundy, 1984], to make several iterations starting with random B and C. If the different iterations converge to the same solution the chance that the solution is a local minimum is small.

Monitoring using PARAFAC The scores for a new batch, a_{new}, are calculated by solving the least squares problem of Eq. 6.54 according to

$$a_{new} = \left[(C \circ B)^T (C \circ B)\right]^{-1} (C \circ B)^T x_{new} \tag{6.58}$$

where $x_{new} \in \mathbb{R}^{KJ}$ are the measurements from the new batch, and C and B are from Eq. 6.54. The residual, e_{new} are calculated according to

$$e_{new} = x_{new} - (C \circ B) a_{new} \tag{6.59}$$

Then T^2 and SPE can be used in the same as for MPCA. For on-line monitoring a multi model approach can be adopted where a different PARAFAC model is built for each time instance.

Dynamic PARAFAC

In [Chen and Yen, 2003] yet another method for batch process monitoring is proposed. The method is based on building a PARAFAC model of a tensor containing time-lagged data much like in BDPCA. The proposed method takes the matrix $X_d^i \in \mathbb{R}^{(K-d)\times(d+1)J}$ in Eq. 6.38 and folds it to the tensor $\underline{X}_d^i \in \mathbb{R}^{(K-d)\times J\times(d+1)}$. Then PARAFAC is used to model each batch at a time according to

$$(X_d^i)^{(K-d)\times(d+1)J} = A^i \left(C^i \circ B^i\right)^T + E^{(K-d)\times(d+1)J} \tag{6.60}$$

Then an overall model is calculated by taking the average over the I batches

$$\begin{aligned} B &= \frac{1}{I}\sum_{i=1}^{I} B^i \\ C &= \frac{1}{I}\sum_{i=1}^{I} C^i \end{aligned} \tag{6.61}$$

One problem not mentioned in the article [Chen and Yen, 2003] is what happens when d is small and when it approaches its maximum. When $d = 0$ the data tensor for one batch becomes $\underline{X}_0^i \in \mathbb{R}^{K\times J\times 1}$, which is a matrix. A similar problem arises when $d = K-1$, the maximum value. $\underline{X}_{K-1}^i \in \mathbb{R}^{1\times J\times K}$.

Alternative Algorithm for DPARAFAC A new way of calculating the model is if instead X_d^i is folded to $\underline{X}_d^i \in \mathbb{R}^{(K-d)\times J\times(d+1)}$ and all batches are stacked on one another to form $\underline{X}_d \in \mathbb{R}^{I(K-d)\times J\times(d+1)}$, see Fig 6.7. Now when $d = 0$ the tensor becomes the unfolded matrix $X^{IK\times J}$ in Fig. 6.4 and could be modeled with PCA. When $d = K - 1$ $\underline{X}_d$ becomes $\underline{X} \in \mathbb{R}^{I\times J\times K}$, which gives the regular PARAFAC model. It can be seen that the delay gives a constraint on how many components that can be used in the model. Interesting issues arise from this but the method is not further analyzed in this thesis.

Monitoring using DPARAFAC Using the new algorithm the calculation of the scores becomes almost the same as for regular PARAFAC. A new batch at time instance k is monitored by calculating the score in the batch mode according to Eq. 6.54 with $x_new = X_d^{new}(k)$ from Eq. 6.38 for the new batch.

Higher-Order Singular Value Decomposition

A method which is related to PARAFAC and Tucker3 is higher-order singular value decomposition, HOSVD, developed in [De Lathauwer *et al.*, 2000a]. The method is a generalization of SVD for matrices to higher-order tensors using tensor notation. The work on HOSVD is briefly described and compared to PARAFAC and Tucker3 in Appendix A.

6.5 Summary

A graphical overview of some of the different multi-way statistical methods is found in Fig. 6.7. The methods are PARAFAC, MPCA on the unfolded matrix $X^{I\times KJ}$, BDPCA using the new alternative algorithm, and DPARAFAC also using the new alternative algorithm.

In this chapter an introduction of the basics of statistical monitoring of processes and a survey of methods for batch process monitoring from the literature have been presented. Two new alternative algorithms for the methods BDPCA and DPARAFAC have been presented. It is shown that depending on the selection of the time lag the methods will turn into the regular MPCA and PARAFAC models. In Chapter 9 the different methods will be implemented for the simulated batch process in Chapter 7 and compared to each other.

There exist even more methods for batch process monitoring based on multivariate statistics than described in this chapter. It is quite hard work just to keep track of them all. Multi-way PLS has been used in various ways for batch process monitoring, see, e.g., [Kourti *et al.*, 1995; Gregersen

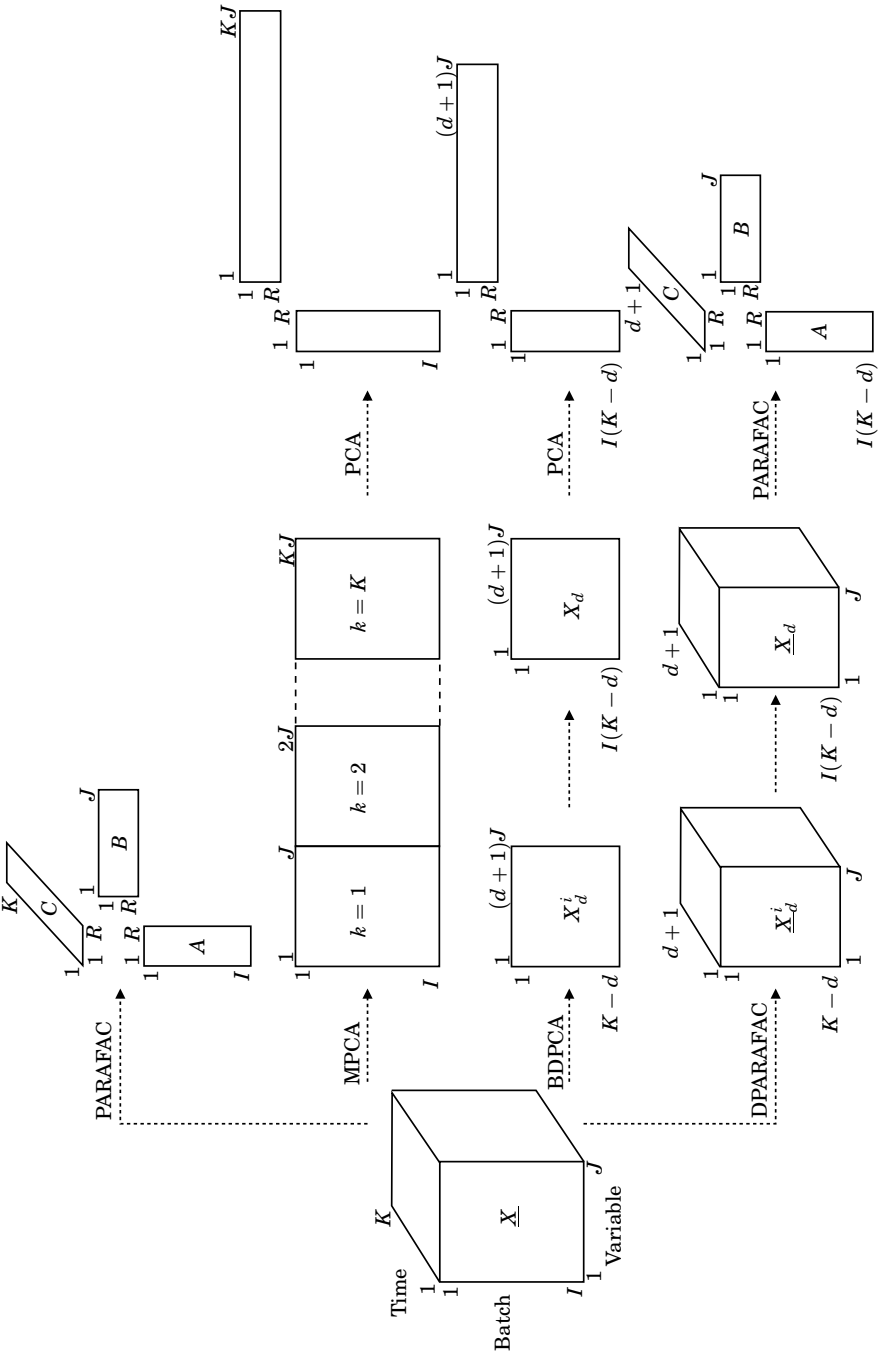

Figure 6.7 Graphical overview of different multi-way statistical methods.

and Jørgensen, 1999]. Multi-way PLS is closely related to batch-wise unfolded MPCA, but the model also describes the relationship between $X^{I\times KJ}$ and the quality measurements organized in the matrix Y. Another method that is closely related to MPCA is multi-way independent component analysis (MICA) [Yoo *et al.*, 2004]. The method is based on independent component analysis (ICA), which searches for a low-dimensional subspace with non-Gaussian components instead of Gaussian as in PCA. The method should therefore provide more meaningful models when the data is non-Gaussian. An introduction and tutorial on ICA is found in [Hyvärinen and Oja, 2000]. Multi-way kernel principal component analysis (MKPCA) [Lee *et al.*, 2004], is based on using kernel PCA (KPCA) [Schölkopf *et al.*, 1998; Choi *et al.*, 2005] to model the data in $X^{I\times KJ}$. KPCA computes principal components in a data set using non-linear kernel functions. In function space analysis based PCA [Chen and Liu, 2001] each process variable in each batch in $X^{I\times KJ}$ is expressed as a polynomial function of time. Then a matrix is formed where each variable is represented by the coefficients of the polynomials and PCA is applied to build a model. Due to constraints in time, none of these methods have been considered in this work.

7

Batch Reactor Simulation Model

7.1 Introduction

To test and compare the different methods described in this thesis a simulation model of a batch reactor is used. The model is based on a batch process model in [Luyben, 1973] and implemented in MATLAB/Simulink. This model has been used as a benchmark problem in several articles considering batch process monitoring, e.g., [Dong and McAvoy, 1995; Wachs and Lewin, 1998; McPherson *et al.*, 2002; Chen and Liu, 2002; Chen and Yen, 2003]. The process consists of two consecutive exothermic first order reactions.

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C \tag{7.1}$$

where k_1 and k_2 are the reaction rates of the first and second reaction. The desired product is B and, thus, the reaction should not go on for too long time, otherwise too much of B will react to C. If stopped early too much of A will be left unreacted. Rapid heating of the tank to a high temperature and then slowly cooling it down will give the optimal yield [Luyben, 1973]. To do this the process is heated with steam and cooled with water by a jacket using two different control valves, V_1 and V_2 in Fig. 7.1. The reference trajectory can be seen in Fig 7.2.

A simplified description of the recipe for the production of one batch is as follows:

- Reactant A is fed to the tank.
- Solvent is added to get the right volume and initial concentration of reactant A.

- The materials are mixed.
- A catalyst is added to the tank to start the reaction.
- Steam is fed to the jacket to heat the tank to the reference temperature as fast as possible.
- The cooling phase is initiated when the specified temperature is reached and cooling water is fed to the jacket to keep the temperature in the tank at the reference trajectory. This is crucial since the reaction is exothermic.
- When the batch has been reacting for the specified duration an inhibitor is added to the batch to stop the reaction.
- The batch is emptied to another tank for further processing.
- Before the next batch can be started the tank needs to be cleaned.

The reactor is equipped with sensors to measure the temperature in the jacket, in the wall between the jacket and the reactor, and inside the reactor. A schematic figure of the process can be found in Fig. 7.1. The measured variables from a typical batch run as well as the control signals can be seen in Fig. 7.2. The concentration profiles of a typical batch are shown in Fig. 7.3.

7.2 Model Equations

The mass and heat balances used in the simulation of the reactor are given by the following equations

$$\begin{aligned}
\frac{dC_A}{dt} &= -k_1 C_A \\
\frac{dC_B}{dt} &= k_1 C_A - k_2 C_B \\
\frac{dT}{dt} &= \frac{-\lambda_1}{\rho C_P} k_1 C_A + \frac{-\lambda_2}{\rho C_P} k_2 C_B - \frac{Q_M}{V \rho C_P} \\
\frac{dT_M}{dt} &= \frac{Q_M - Q_J}{\rho_M C_M V_M} \\
Q_M &= h_i A_i \left(T - T_M\right)
\end{aligned} \tag{7.2}$$

where C_A and C_B are the concentrations of A and B in the reactor, k_1 and k_2 are the reaction rates given by the Arrhenius expression

$$k_j = \alpha_j \exp\left(\frac{-E_j}{R(T + 460)}\right) \tag{7.3}$$

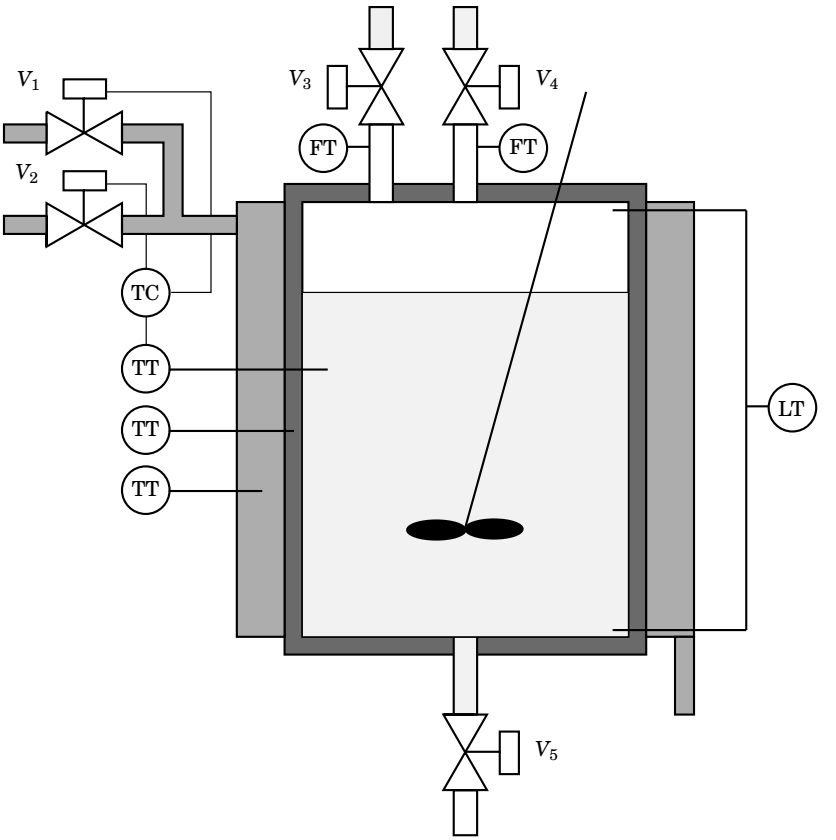

Figure 7.1 Schematic figure over the simulated batch process.

where α_j is a constant and E_j is the activation energy for the two reactions, $j = 1, 2$. R is the universal gas constant and T is the absolute temperature in the reactor. λ_1 and λ_2 are the heat of reaction, ρ and ρ_M are the densities of the contents of the reactor and the material of the wall respectively, and C_P and C_M are the heat capacities of the contents and the wall respectively. A_i is the heat transfer area between the reactor and the wall, h_i is the heat transfer coefficient, and V and V_M are the volumes of the contents and the wall. T_M is the temperature in the wall between the reactor and the jacket, Q_M is the heat transferred from the reactor to the wall, T_J is the temperature in the jacket, and Q_J is the heat transferred from the jacket to the wall.

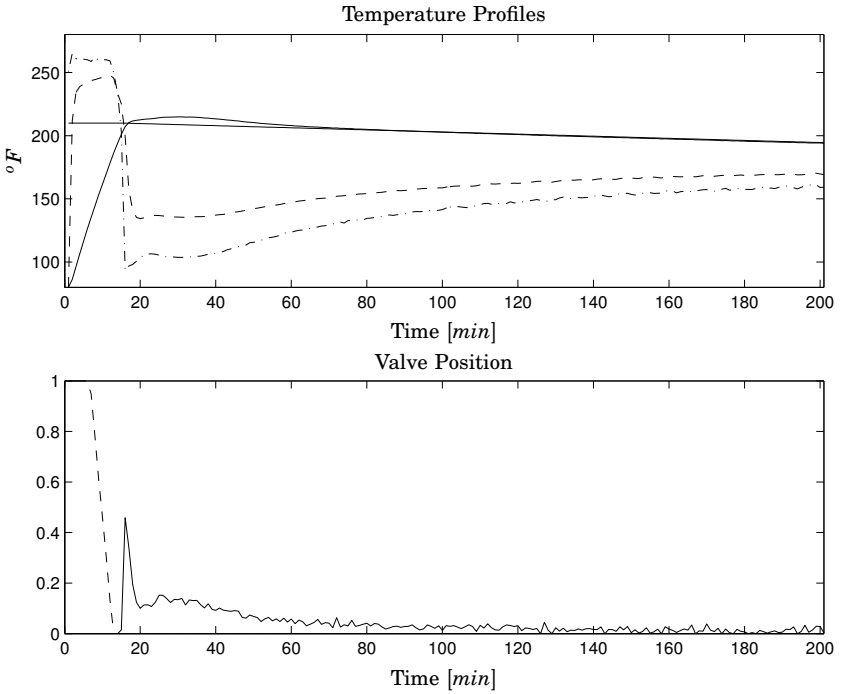

Figure 7.2 Measurements from a typical batch. The upper plot shows the temperature profiles: inner reactor temperature - solid line, wall temperature - dashed line, and jacket temperature - dash-dotted line. Also shown is the reference trajectory for the reactor temperature. The lower plot shows the heating (dashed line) and cooling valve (solid line) positions.

The equations for the jacket are different depending on the media in the jacket. During the heating with steam they are given by

$$\begin{aligned}
\frac{d\rho_s}{dt} &= \frac{w_s - w_c}{V_J} \\
\rho_s &= \frac{M_w P_J}{R\,(T_J + 460)} \\
P_J &= \exp\left(\frac{A_{vp}}{T_J + 460} + B_{vp}\right) \\
w_s &= X_s C_{Vs}\sqrt{P_{S0} - P_J} \\
Q_J &= -h_{os} A_o\,(T_J - T_M) \\
w_c &= -\frac{Q_J}{H_s - h_c}
\end{aligned} \tag{7.4}$$

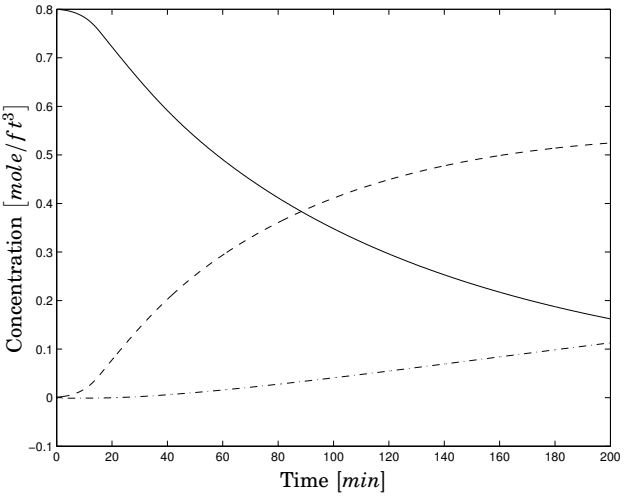

Figure 7.3 Concentration profiles of a typical batch: Concentration of A (C_A) - solid line, Concentration of B (C_B) - dashed line, and Concentration of C (C_C) - dash-dotted line.

where ρ_s is the density of the steam, w_s is the steam coming into the jacket, w_c is the rate of condensation in the jacket, V_J is the volume of the jacket, M_w is the molecular weight of water, P_J is the pressure in the jacket, P_{S0} is the pressure of the steam coming into the jacket, H_S is the enthalpy of the steam coming into the jacket, and h_c is the enthalpy of the condensate leaving the jacket. X_s is the control signal to the steam valve V_1, A_o is the heat transfer area between the jacket and the wall, and h_{os} is the heat transfer coefficient of the steam. The equations for the jacket during the heating are numerically solved for the absolute temperature in the jacket T_J in the simulation.

During the filling of the jacket after stopping the heating and starting the cooling phase the equations for the jacket are given by

$$\begin{aligned} A_o &= V_J \frac{A_o^{tot}}{V_J^{tot}} \\ \frac{dV_J}{dt} &= F_w \\ \frac{dV_J T_J}{dt} &= F_w T_{J0} + \frac{Q_J}{\rho_J C_J} \\ Q_J &= h_{ow} A_o \left(T_M - T_J\right) \\ F_w &= C_{Vw} X_w \sqrt{P_{w0}} \end{aligned} \tag{7.5}$$

Table 7.1 Simulation parameters

α_1	729.55	min^{-1}	V_J^{tot}	18.83	ft^3
α_2	6567.6	min^{-1}	V	42.5	ft^3
E_1	15,000	Btu/mole	V_M	9.42	ft^3
E_2	20,000	Btu/mole	λ_1	–40,000	Btu/mole
A_{vp}	–8744.4	^{o}R	λ_2	–50,000	Btu/mole
B_{vp}	15.70		P_{S0}	35	psi
T_0	80	^{o}F	C_p	1	$\text{Btu/lb}_m\text{-}^o\text{R}$
T_{J0}	80	^{o}F	C_M	0.12	$\text{Btu/lb}_m\text{-}^o\text{R}$
C_{Vs}	112	$\text{lb}_m/\text{min-psi}^{1/2}$	C_J	1	$\text{Btu/lb}_m\text{-}^o\text{R}$
C_{Vw}	100	$\text{gal/min-psi}^{1/2}$	ρ	50	lb_m/ft^3
h_{os}	1000	$\text{Btu/hr-}^o\text{R-ft}^2$	ρ_M	512	lb_m/ft^3
h_{ow}	400	$\text{Btu/hr-}^o\text{R-ft}^2$	ρ_J	62.3	lb_m/ft^3
h_i	160	$\text{Btu/hr-}^o\text{R-ft}^2$	R	1.987	$\text{Btu/mole-}^o\text{R}$
A_o^{tot}	56.5	ft^2	C_{A0}	0.80	mole/ft^3
A_i	56.5	ft^2	$H_s - h_c$	939	Btu/lb_m

where A_o is the part of the heat transfer area used for cooling, A_o^{tot} is the total heat transfer area when the jacket is full. V_J is the volume of cooling water in the jacket and V_J^{tot} is the total volume. F_w is the flow of cooling water, X_w is control signal to the cooling water valve V_2. T_{J0} is the cooling water temperature and P_{w0} is the cooling water temperature and pressure at the inlet. The dynamics for the jacket when it is full of water is given by

$$\frac{dT_J}{dt} = \frac{F_w}{V_J^{tot}}(T_{J0} - T_J) + \frac{Q_J}{C_J V_J^{tot} \rho_J} \tag{7.6}$$

The values of the parameters used in the simulations are given in Table 7.1.

7.3 Controller

The temperature in the reactor T is controlled by a discrete-time PID controller with different parameters depending on if the batch is in the heating or the cooling phase. The controller is implemented according

to [Wittenmark *et al.*, 2000]

$$
\begin{aligned}
P(k) &= K_c\left(\beta T_{ref}(k) - T(k)\right) \\
D(k) &= \frac{T_d}{T_d + Nh} D(k-1) - \frac{K_c T_d N}{Td + Nh}\left(T(k) - T(k-1)\right) \\
v(k) &= P(k) + I(k) + D(k) \\
u(k) &= \begin{cases} 1, & v(k) \geq 1 \\ v(k), & 0 < v(k) < 1 \\ 0, & v(k) \leq 0 \end{cases} \\
I(k+1) &= I(k) + \frac{hK_c}{T_i}\left(T_{ref}(k) - T(k)\right) + \frac{h}{T_t}\left(u(k) - v(k)\right)
\end{aligned}
\tag{7.7}
$$

where K_c is the controller gain, N limits the gain for high frequencies, T_d is the derivative gain, T_i is the integral time, T_t is tracking time constant, and β is the set point weighting. The sampling time in the controller, h, was $12s$. The values of the controller parameters are given in Table 7.2. The parameters changes when the batch changes phase from heating to cooling. There is no integral action during the heating phase.

Table 7.2 Controller parameters

Parameter	Heating	Cooling
K_c	0.02	-0.02
T_d	3	5
N	10	10
T_i	∞	100
T_t	∞	1000
β	1	1

7.4 Recipe Implementation

One of the recipes used for the production of a batch of Product B has been implemented in JGrafchart according to Fig. 7.4. The recipe described in the beginning of the chapter is implemented using procedure steps that call equipment operations, also implemented in JGrafchart, according to Fig. 3.5.

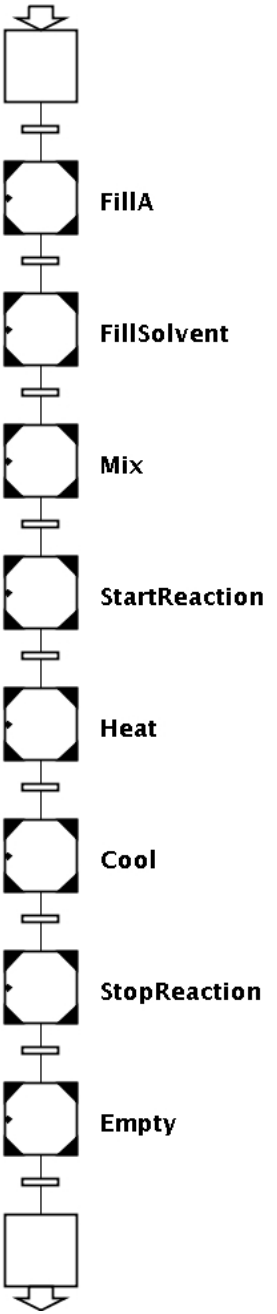

Figure 7.4 Recipe for the production of one batch of the simulated process.

7.5 Definitions of Faults

To be able to detect faults it is important to specify which faults may occur and when they may occur during the execution of a control recipe. The type of faults that may occur and the level of model complexity available for the process to be monitored will influence the choice of method to be used.

Usually the first thing to do when designing a fault diagnosis system is to define the faults that may occur in the process, when they may occur, and how the faults may influence the process. Several different methods can be used for this, e.g., FMEA or Hazop analysis described briefly in Chapter 5.

Faults that may be considered for the different operations during the manufacturing of a batch in the simulated batch process in are defined below. The definitions of the faults are structured according to the recipe described in Fig. 7.4 and the process equipment seen in Fig. 7.1.

- Filling of A
  - Bias in the flow meter
    - Primary effect: Wrong amount of A added to the tank.
    - Secondary effect: Wrong C_{A0}
  - Bias in the level sensor
    - Primary effect: Wrong amount of A added to the tank.
    - Secondary effect: Wrong C_{A0} and V.
  - V_3 is stuck closed
    - Primary effect: No flow of A.
    - Secondary effect: Delay of recipe execution.
  - Leakage in the valve V_5.
    - Primary effect: Waste of A. Risk to personnel.
    - Secondary effect: Low C_{A0}.
  - V_3 is stuck open at the end of filling
    - Primary effect: Too much A added.
    - Secondary effect: A need to be emptied out, high C_{A0}, or high V.
- Filling of solvent
  - Bias in the flow meter
    - Primary effect: Wrong amount of solvent added to the tank.

    - Secondary effect: Wrong C_{A0} and V.
  - Bias in the level sensor
    - Primary effect: Wrong amount of solvent added to the tank.
    - Secondary effect: Wrong C_{A0} and V.
  - V_4 is stuck closed
    - Primary effect: No flow of solvent.
    - Secondary effect: Delay of recipe execution.
  - Leakage in the valve V_5
    - Primary effect: Waste of A and solvent. Risk to personnel.
    - Secondary effect: Low C_{A0} since extra solvent is added during filling.
  - V_4 is stuck open at end
    - Primary effect: Too much solvent added.
    - Secondary effect: Low C_{A0}.
- Mixing
  - Poor mixing
    - Primary effect: Lower reaction rate.
    - Secondary effect: -
  - Leakage in the valve V_5
    - Primary effect: Waste of A and solvent. Low V.
    - Secondary effect: -
- Adding of catalyst
  - Poor catalyst
    - Primary effect: Higher activation energies, E_1 and E_2.
    - Secondary effect: -
  - Leakage in the valve V_5
    - Primary effect: Waste of A and solvent. Low V.
    - Secondary effect: -
- Heating
  - Low heat transfer coefficient, h_i, due to fouling
    - Primary effect: Slower heating.
    - Secondary effect: Higher control signal.
  - Low steam pressure, low steam temperature

    - Primary effect: Slower heating.
    - Secondary effect: -
  - Leakage in the valve V_5
    - Primary effect: Waste of A and solvent. Low V.
    - Secondary effect: -
  - Leakage in the jacket
    - Primary effect: Less heating. Risk to personnel.
    - Secondary effect: Higher control signal.
  - Unwanted reaction pathway
    - Primary effect: Less yield of the wanted product.
    - Secondary effect: Product off specification.
- Cooling
  - Low heat transfer coefficient, h_i, due to fouling
    - Primary effect: Less cooling.
    - Secondary effect: Higher control signal. Runaway reaction.
  - High temperature in cooling water
    - Primary effect: Less cooling.
    - Secondary effect: Higher control signal. Runaway reaction.
  - Leakage in the valve V_5
    - Primary effect: Waste of A and solvent. Low V.
    - Secondary effect: -
  - Leakage in the jacket
    - Primary effect: Less cooling.
    - Secondary effect: Higher control signal.
  - Unwanted reaction pathway
    - Primary effect: Less yield of the wanted product.
    - Secondary effect: Product off specification.
- Adding of inhibitant
  - Reaction not terminated
    - Primary effect: Reaction time too long. Less yield.
    - Secondary effect: Cooling needs to continue.
  - Leakage in the valve V_5
    - Primary effect: Waste of A and solvent. Low V.

    - Secondary effect: -

- Emptying
  - V_5 is stuck closed
    - Primary effect: No flow of product.
    - Secondary effect: Delay of recipe execution.

A large amount of faults may occur that are associated with the reactions. Looking at the dynamic model in Eq. 7.2 it can be seen that the reactions depend on a number of parameters, such as the activation energies, that may vary between batches or during a batch. One way of systematically determine how the parameters affect the system is the diagnostic model processor, DMP, method, which is applied to the batch process in Chapter 8.

A thorough list of expected faults and exceptions associated with a process will make the design of the diagnosis system more structured. This gives an indication of which faults are most severe and most important to be able to handle, and where resources should be distributed used during the development of the system. For example, a high or low initial concentration of A will have a large impact on the yield of the final product, while an increase of the inner heat transfer coefficient due to fouling does not affect the reaction as long as the controller is able to control the temperature in the reactor. Instead it is the control signals, the heating and the cooling flow, that will be influenced. An increase of the activation energy for reaction one will lower the amount of A reacting to B, and thus the yield of the product, if a fixed time is used for each batch. An increase of the initial concentration of A will increase the energy release at the beginning of the batch and may even lead to a run-away reaction if the cooling is not designed for such an event.

Some of the faults listed here will be considered in the implementations and comparisons of the different methods in the following chapters.

7.6 Summary

A benchmark model have been implemented that will be used in the subsequent chapters to compare methods and discuss the properties of a batch process. A simulation model makes it possible to simulate normal behavior as well as the influence of faults, i.e., fault signatures. A model gives a deep knowledge about the process and can be used at different stages for designing both the control and the diagnostic system.

8

Monitoring of Batch Processes using Diagnostic Model Processor and Deep Model Algorithm

8.1 Introduction

In this chapter the two methods diagnostic model processor, DMP, and deep model algorithm, DMA, are described. They were originally developed for fault detection in continuous processes and here they are applied to monitoring of faults in the simulated batch described in Chapter 7. The underlying assumptions of these methods are that the batch process is either operating at normal conditions where all parameters have their nominal values and only noise is added to the measurements or that one specific fault at the time is acting on the process.

8.2 Diagnostic Model Processor

The diagnostic model processor method, DMP, was developed at the University of Delaware, see [Petti, 1992], for model-based process fault diagnosis of continuous processes. DMP is based on using model equations in a knowledge-based manner and is closely related to governing equations [Kramer, 1987], the method of minimal evidence [Fickelscherer, 1990], and parity relations [Gertler, 1998]. By examining the direction and level of violation of the model equations and looking at the assumptions,

i.e., the fault possibilities that can be diagnosed, on which they depend, the most likely fault is determined. The relations, i.e., the sensitivities and tolerances, between the equations and the assumptions do not need to be fixed.

A model equation e_m written on residual form is given by

$$e_m = C_m(P, a) \tag{8.1}$$

where C_m is the mth model equation rewritten on residual form, P are the measurements from the process, and a are the modeling assumptions. The assumptions can, e.g., be that there is no fouling on the heat transfer areas or that a tank is not leaking. By determining the assumptions that are not valid, the faults are identified.

An example of a model equation is the volume balance for a tank

$$\frac{dV(t)}{dt} = F_{in}(t) - F_{out}(t) \tag{8.2}$$

which rewritten on discretized residual form becomes

$$e_1 = L(k) - L(k-1) - \frac{h}{A}\left[F_{in}(k-1) - F_{out}(k-1)\right] \tag{8.3}$$

where V is the volume of the tank, $L(k)$ is the measured level in the tank at sample k, h is the sample time, A is the area of the tank, and $F_{in}(k)$ and $F_{out}(k)$ are the flows in and out to the tank at sample k.

Associated with each model equation are upper and lower *tolerances*, τ_H and τ_L, which represent the high and low limits for which the process is considered to operate in a normal state. The selection of τ is not of critical importance since DMP uses a fuzzy set to classify if the equation is violated. The tolerance is also used when calculating the sensitivity of the model equation to an assumption. For each model equation a set of assumptions is developed which if satisfied guarantee the satisfaction of the equation.

Since the residuals in e have different magnitude a satisfaction vector is constructed using the tolerances τ. Each *satisfaction value* sf_m is calculated according to

$$sf_m = \frac{(e_m/\tau_m)^\eta}{1 + (e_m/\tau_m)^\eta}\text{sign}(e_m) \tag{8.4}$$

where $\text{sign}(e_m)$ is the sign of the residual e_m. If e_m is positive it gives sf_m a positive sign and vice versa. The function sf_m is a sigmoidal function between -1 and 1, where the steepness of the function is given by the parameter η, see Fig. 8.1.

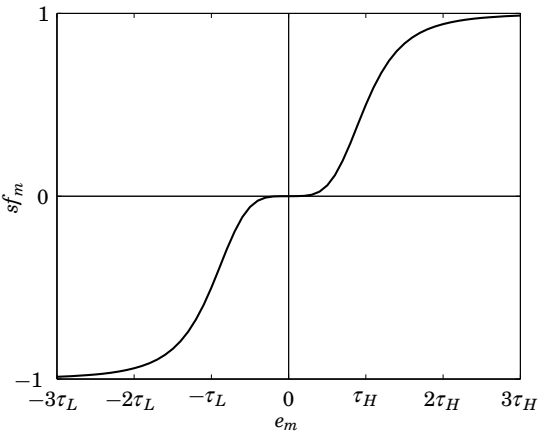

Figure 8.1 Satisfaction value calculated using the sigmoidal function in Eq. 8.4, $\eta = 4$.

Different faults may give rise to different magnitudes in the same residual. Therefore a matrix of *sensitivity values*, S, which describes the relationship between the residual equation e_m and the assumption a_n is calculated. Where a partial derivative of the residual can be calculated the sensitivity value is calculated according to

$$S_{mn} = \frac{\partial C_m}{\partial a_n} \Big/ |\tau_m| \tag{8.5}$$

The larger the partial derivative of the equation is with respect to the assumption, the more sensitive the deviation is to the assumption. If the assumption is implicit (i.e., it is not directly related to the variables or parameters) or no derivative can be calculated the magnitude of the sensitivity may be given an arbitrary value according to experience, often a magnitude of 1 is used. The sensitivity matrix is calculated on-line during operation.

The determination if the state of the process is normal is based on combining the satisfaction vector sf_m with the sensitivity matrix S to a vector of failure likelihoods, F. The *failure likelihood* based on the arithmetic average of the weighted evidence for assumption (fault) n is given by

$$F_n = \frac{\sum_{m=1}^{M} S_{mn} sf_m}{\sum_{m=1}^{M} |S_{mn}|} \tag{8.6}$$

where M is the total number of modeling equations. The satisfaction values that are most sensitive to fault assumption n are weighted higher

than those that are less sensitive. The failure likelihoods do not consider the satisfaction values that are independent of an assumption, i.e., where the sensitivity S_{mn} is equal to zero. The reason for this is to be able to detect faults of small magnitude and not discount fault possibilities at an early stage.

Supportability [Kramer, 1987] is a method for determining if a specific single fault has occurred. The supportability of the nth fault, q_n, can be added to DMP using the equation

$$q_n = \left[\prod_{m=1}^{r} |sf_m|\right]\left[\prod_{m=r+1}^{M} 1 - |sf_m|\right] \tag{8.7}$$

where the order of the satisfaction values have been changed so that the r first depend on the assumption a_n and the last $M - r$ do not depend on the assumption a_n. The supportability can also be calculated for the case of the assumption that no fault has occurred by letting $r = 0$. A more detailed approach can also be taken by including the direction of the satisfaction values.

The success of DMP in detecting and isolating different faults is dependent on the available model equations. An analysis tool to determine if different faults give the same residuals is what is known as the comparison matrix in DMP. The comparison matrix is based on the *significance* of each model equation in the diagnosis of each fault. The significance for model equation m to fault n is calculated as

$$sig_{mn} = \frac{S_{mn}}{\sum_{m=1}^{M} |S_{mn}|} \tag{8.8}$$

A *comparison matrix* can also used to simulate if different faults may give rise to the same residuals and thus make it impossible to distinguish between the faults. The comparison matrix for each assumption a_n is calculated by

$$Comp_{nl} = \sum_{m=1}^{M} \frac{sig_{mn} \cdot sig_{lm}}{\max\left(|sig_n|\right)} \tag{8.9}$$

where $\max\left(|sig_n|\right)$ is the maximum absolute value for the significance of assumption a_n. A large value of $Comp_{nl}$ indicates that the lth fault can be signaled as the fault even though fault n is the origin. These simulations point out shortcomings of the diagnosis system and be used as a basis to, e.g., add more sensors or use a more detailed model to be able to distinguish between the two faults.

Problems with DMP

One problem arises if two different faults affect the same residual but the magnitudes of the residuals are very different. How should the tolerance for the residual be chosen? If it is chosen to fit the fault giving rise to a small residual, thus having a small value, the sensitivity to the other fault will be very large. If it is chosen to fit the fault giving rise to a large residual the residual will not give a contribution to the satisfaction when the other fault occurs.

As described above the method chooses a tolerance specific only to the residual, calculates the satisfaction, and then weights the satisfaction with the sensitivity. If instead the sensitivity is used before the satisfaction is calculated, or in other words the tolerance is specific for each residual and fault, this problem is circumvented. Now a matrix consisting of the tolerances τ_{mn} is used to calculate the satisfaction value. In the next section a method using a tolerance specific to each residual and fault is described. This could also be used in the DMP method to calculate the failure likelihood.

8.3 Deep Model Algorithm

The deep model algorithm DMA [Chang *et al.*, 1994] is a method strongly related to DMP. In DMA a vector of satisfaction values, one for each fault, is calculated, i.e., $sf_i = [sf_{1n}\ sf_{2n}\ \dots\ sf_{mn}\ ]^T$ for each residual. If the residual e_m is dependent on the fault a_n then the satisfaction value is calculated as

$$sf_{mn} = \frac{(e_m/\tau_{mn})^\eta}{1+(e_m/\tau_{mn})^\eta}\text{sign}(e_m/\tau_{mn}) \tag{8.10}$$

If the residual is independent of the fault the satisfaction value is instead calculated as

$$sf_{mn} = \left[1 - \frac{(e_m/\tau_{m,min})^\eta}{1+(e_m/\tau_{m,min})^\eta}\right] sign^* \tag{8.11}$$

where

$$\tau_{m,min} = min(|\tau_{mr}|) \tag{8.12}$$

with τ_{mr} being the tolerances in the calculation where the residual is dependent of the fault. This way if the residual is of such a magnitude that it gives a high satisfaction for another fault the satisfaction for independent faults will decrease. The sign of the satisfaction value when it is independent of the fault is given by

$$sign^* = \begin{cases} \text{sign}(sf_n^*) & if\ |sf_n^*| \geq \widehat{sf}_n \\ 1 & if\ |sf_n^*| < \widehat{sf}_n \end{cases} \tag{8.13}$$

where sf_n^* is the largest (in absolute sense) value of the satisfaction values dependent of the nth fault. The reason mentioned in [Chang *et al.*, 1994] for using the threshold $\widehat{sf}_n$ is to avoid that measurement noise changes the sign of the satisfaction value. Another reason is that the next calculated index, the *degree of fault*, d_n, is not equal to zero even though the residuals are zero. The degree of fault is defined as

$$d_n = \frac{1}{M} \sum_{m=1}^{M} sf_{mn} \tag{8.14}$$

The satisfaction value for a residual that is independent of the fault is equal to unity when e is a zero vector. If R residuals are dependent on a fault a_n, then the value of d_n will be $\frac{M-R}{M}$. Choosing $\widehat{sf}_n$ too low can make d_n flicker between plus and minus this value. If the number of residuals M is large and R is small d_n will flicker between approximately plus and minus 1. In this case not much information is found in the degree of fault. One more index is suggested in the method, the *consistency factor*, cf_n, defined by

$$cf_n = 1 - \left[\frac{sf_{max,n} - sf_{min,n}}{\max\left(|sf_{1n}|,\ |sf_{2n}|,\ \ldots,\ |sf_{Mn}|\right)} \right] \tag{8.15}$$

where $sf_{max,n}$ and $sf_{min,n}$ are the largest and smallest satisfaction values for the nth fault. The consistency factor tells if all the satisfaction values are close to each other or not. If all the satisfaction values are close to each other the consistency factor is close to 1 meaning that the fault is probable. If the consistency factor is close to -1 the fault is an unlikely cause of the problem.

Problems with DMA

A problem arises with the sign of the satisfaction value when a residual is independent of a fault, described in Eq. 8.13. If sf_n^* has a value around $-\widehat{sf}_n$, $sign^*$ will flicker between ± 1. A better approach would be to instead use a dead-band with the value $\pm\widehat{sf}_n$. This would avoid the flickering.

Another problem which is related to the problem above is when a fault only affects few of the residuals. This gives the degree of fault for this fault a large value even though all the residuals are equal to zero. The consistency factor would be equal to zero and that should also be taken into consideration. If the residual which is affected by the fault approaches $-\widehat{sf}_n$ and stays around this value the degree of fault will flicker between two large in magnitude values. If for example the fault only affects one of 10 residuals and this residual gives a satisfaction with values around,

e.g., $-sf_n^* = -0.1$ the degree of fault would approximately flicker between $(1 \cdot 9 - 0.1)/10 = 0.89$ and $(-1 \cdot 9 - 0.1)/10 = -0.91$. A plot of such a flickering would hardly make an operator positive to using the diagnosis system.

8.4 Applying DMP and DMA for Batch Process Monitoring

Different residuals can be calculated depending on the complexity of the model equations and other information available. Here the residuals for the simulated batch process will be stated depending on the information available.

During the production of a batch some faults can only occur at certain parts of the recipe, e.g., the initial concentration of A, C_{A0}, can only be influenced during the filling of A and of the solvent. Therefore the calculation of the residuals for the batch is dependent on which phase is active at a specific moment. There also exist faults that may occur during any part of the recipe, e.g., leakage through the valve at the bottom of the reactor, V_5. Even though a fault may occur during the whole batch the fault detection is dependent on the phase being active in the recipe, e.g., the detection logic for the above mentioned leak depend on if something is being added to the reactor or not. The residuals will be stated for every phase, but the monitoring will be concentrated to the cooling phase.

Nominal Design Trajectories

The lowest level of model complexity is when only the nominal reference trajectory of the controlled variable T is available. Let us go one step further and assume that the nominal trajectories of all the measured variables are available in the diagnostic system. Knowing the nominal design trajectories the following residuals can be calculated for the different phases of the recipe.

Filling of A During the filling of A the level in the reactor should increase according to the amount of A being added to the reactor. The measurements of both the flow into the reactor and the level in the reactor make it possible to state the following residual

$$e_1(k_A) = L(k_A) - L(k_A - 1) + hF_A(k_A - 1) \tag{8.16}$$

where $L(k_A)$ and $F_A(k_A)$ are the level measurement and the flow measurement of A at k_A samples of running the phase, and h is the sampling time in the diagnosis system. The level in the reactor should be equal to zero at the beginning of the phase, i.e., $L(0) = 0$. This residual can be

associated with several different faults, e.g., leakage through V_5, bias in F_A, or bias in L.

If the valve for the adding of A is an on/off valve the following residual can be used to determine if the valve is starting to get clogged or if it has got stuck in the closed position.

$$e_2(k_A) = F_A(k_A) - \overline{F}_A \tag{8.17}$$

where $\overline{F}_A$ is the nominal flow through the valve.

Since the phase is repeated every batch, a residual can be calculated for the level according to

$$e_3(k_A) = L(k_A) - \overline{L}(k_A) \tag{8.18}$$

where $\overline{L}(k_A)$ is the nominal level at time k_A. The total duration of the phase can be used to update the nominal level and the nominal duration of the phase used in the scheduler

Filling of Solvent During the filling of solvent the level in the reactor should increase according to the amount of solvent being added to the reactor. Again, measurements of the flow into the reactor and the level in the reactor make it possible to state the following residual

$$e_1(k_S) = L(k_S) - L(k_S - 1) + hF_S(k_S - 1) \tag{8.19}$$

where $F_S(k_S)$ is the flow measurement of solvent at k_S samples of running this phase. The level in the reactor at the beginning of the phase should be equal to the level at the end of the phase for filling A, i.e., $L(k_S = 0) = L(k_A^{end})$.

If the valve for the adding of solvent is an on/off valve the following residual can be used to determine if the valve is starting to get clogged or if it is stuck in the closed position.

$$e_2(k_S) = F_S(k_S) - \overline{F}_S \tag{8.20}$$

where $\overline{F}_S$ is the nominal flow through the valve.

Again, since the phase is repeated every batch, a residual can also be calculated for the level according to

$$e_3(k_S) = L(k_S) - \overline{L}(k_S) \tag{8.21}$$

Mixing During the mixing nothing is added or taken out of the reactor but still a residual for the level could be calculated to detect leakage through the out-valve, V_5, using

$$e_1(k_M) = L(k_M) - L(k_M = 0) \tag{8.22}$$

where $L(0)$ is the level at the beginning of the mixing phase and at the of the filling of solvent, i.e., $L(k_M = 0) = L(k_S^{end})$.

Start of Reaction In the simulation model there is no real initiation of the reaction and the state in the recipe can be considered as a wait step. The residual for the leakage through the out-valve, V_5, can be implemented in the same way as in the mixing phase, see Eq. 8.22.

Heating During the heating phase the deviation from the nominal trajectories can be calculated for the reactor temperature, T, the wall temperature, T_M, the jacket temperature, T_J, and the position of the steam-valve V_1, X_S. The residuals are calculated by the following equations.

$$e_1(k_H) = T(k_H) - \overline{T}(k_H) \tag{8.23}$$

$$e_2(k_H) = T_M(k_H) - \overline{T}_M(k_H) \tag{8.24}$$

$$e_3(k_H) = T_J(k_H) - \overline{T}_J(k_H) \tag{8.25}$$

$$e_4(k_H) = X_S(k_H) - \overline{X}_S(k_H) \tag{8.26}$$

where k_H is the number of samples since the heating phase was started.

Another important information for the fault detection is when the switch from the heating phase to the cooling phase will occur. The switch can take place earlier or later than the nominal switching. An earlier switching indicates that the temperature in the reactor has reached the switching temperature earlier than the nominal time. This will happen when the time derivative of T is higher than normal during the phase, e.g., if the initial concentration of A, C_{A0}, is higher than nominal, or if the temperature sensor of T has a bias. Without a reaction model it is not clear how a bias in the temperature sensor of T will affect the switching time. A positive bias will make the switching logic detect that the switching threshold has been reached even though the real temperature is below. On the other hand the controller will control the temperature in the reactor so that the real temperature is below the nominal temperature, which will decrease the rate of reaction. The opposite is true for a negative bias.

A late switching between the phases may be due to a low C_{A0}. It can also be the result of fouling which will decrease the heat transfer parameter, h_i, and thus the heating of the reactor will decrease. An increase in the activation energies, E_1 and E_2, due to, e.g., impurities in the reactant, will decrease the reaction rate and thus the rate of increase of the reactor temperature T. Again the residual for the leakage through the out-valve, V_5, can be implemented in the same way as in the mixing phase, see Eq. 8.22.

Cooling During the cooling phase the residuals for the deviation from the nominal trajectories can be calculated similarly to the residuals during

the heating phase.

$$e_1(k_C) = T(k_C) - \overline{T}(k_C) \tag{8.27}$$
$$e_2(k_C) = T_M(k_C) - \overline{T}_M(k_C) \tag{8.28}$$
$$e_3(k_C) = T_J(k_C) - \overline{T}_J(k_C) \tag{8.29}$$
$$e_4(k_C) = X_W(k_C) - \overline{X}_W(k_C) \tag{8.30}$$

where X_W is the position of the cooling-water valve, V_2, and k_C is the number of samples since the cooling phase was started.

Again the residual for the leakage through the out-valve V_5 can be implemented in the same way as in the mixing phase, see Eq. 8.22.

Stop of Reaction In the simulation model there is no real stopping of the reaction and the state in the recipe can be considered as a wait step. The residual for the leakage through the out-valve V_5 can be implemented in the same way as in the mixing phase, see Eq. 8.22.

Empty In this phase the product in the reactor is emptied to further processing in another unit. If there is no flow meter for the out-flow a residual for the emptying can be calculated using a nominal value for the out-flow, $\overline{F}_E$.

$$e_1(k_C) = L(k_C) - L(k_C - 1) - h\overline{F}_E \tag{8.31}$$

If the nominal level profile, $\overline{L}(k_C)$, is known from earlier batches the residual can calculated as

$$e_2(k_C) = L(k_C) - \overline{L}(k_C) \tag{8.32}$$

Reaction Model

A model of the reaction in the batch process can be utilized in several ways. The model makes it possible to calculate more residuals to use in DMP/DMA.

In [Petti, 1992] it was suggested to compare an estimation of the time derivative of the measured variables and the calculated value of the time derivative from the reactor model, i.e., the right hand side of the differential equations in the model, using the current measurements and the nominal parameters. This kind of residual is highly sensitive to noise since both the derivative estimated with a difference approximation and the right hand side of the differential equations in the model are calculated from measurements.

For a batch process the behavior of a nominal batch is known from the design of the process. This can be used for the calculation of more

residuals. One is the difference between a difference approximation of the derivative and the time derivative calculated from the model for a nominal batch. Another residual is the difference between the calculated value of the time derivative from the reactor model for the batch in progress and the time derivative from the reactor model for a nominal batch. It turns out that these residuals are less sensitive to noise than the suggestion from [Petti, 1992] and still contains information about the process. In the rest of this section the following notation will be used, $T(k)$ is a measurement of the variable T at time k, $\dot{T}(k)$ is a time derivative of T calculated from the model using the nominal parameters and measurements from time k, and $\overline{\dot{T}}(k)$ is a time derivative of T calculated from the model for a nominal batch.

To be able to use the model equations in the benchmark simulated batch process model in Chapter 7 some estimation of the unmeasured states C_A and C_B must be used. The model might make it possible to estimate the state of the process using an Extended Kalman Filter EKF, see, e.g., [Dochain, 2003]. For example, if the concentrations of A and B can be estimated during the heating phase this could be used in the cooling phase. If this is not possible the best estimation of the concentrations available are the nominal trajectories from the design of the process, which are used here. More on the use of EKF for batch processes is found in Chapter 10.

Using the suggested methods for structuring of exception handling in Chapter 3 makes it possible to use different fault diagnosis strategies during the different operations or phases in the recipe. For example, during the filling of reactant A and the solvent the amount that has been added to the vessel is very important to monitor but after these operations the initial concentration cannot be influenced. Since the diagnosis during these operations might fail to detect an abnormality in the initial concentration it is important to keep monitoring for this fault during the heating operation but now using another strategy specific to the heating operation. If no deviation in the initial concentration of A has been detected after a certain amount of time into the batch the monitoring for this fault should probably be stopped and ruled out as a candidate when an abnormal situation arises after this point of time.

Heating Looking at the simulation model described in Chapter 7, which is considered to be the real plant, it can be concluded that trying to model the steam condensate system in the jacket is not an easy task. Therefore, during the heating phase only the dynamic model equations for the reactor temperature and the wall temperature are used in the residuals. The new

residuals which could be used are

$$e_5(k_H) = T(k_H) - T(k_H - 1) - h\dot{T}(k_H - 1) \tag{8.33}$$
$$e_6(k_H) = T_M(k_H) - T_M(k_H - 1) - h\dot{T}_M(k_H - 1) \tag{8.34}$$
$$e_7(k_H) = T(k_H) - T(k_H - 1) - h\dot{\overline{T}}(k_H - 1) \tag{8.35}$$
$$e_8(k_H) = T_M(k_H) - T_M(k_H - 1) - h\dot{\overline{T}}_M(k_H - 1) \tag{8.36}$$
$$e_9(k_H) = \dot{T}(k_H) - \dot{\overline{T}}(k_H) \tag{8.37}$$
$$e_{10}(k_H) = \dot{T}_M(k_H) - \dot{\overline{T}}_M(k_H) \tag{8.38}$$

where $\dot{T}$ and $\dot{T}_M$ are the time derivatives calculated from the model with nominal parameters and measurements, and $\dot{\overline{T}}$ and $\dot{\overline{T}}_M$ are the time derivatives from the model for a nominal batch, at sample k_H.

Cooling At the beginning of the cooling phase the steam from the heating condensates and the jacket is filled with cooling water. This part is also hard to model and it is assumed that no model of this is available. Once the jacket is filled with cooling water the model for the jacket becomes much simpler and the dynamic equation describing the jacket can be incorporated among the residuals, i.e.,

$$e_5(k_C) = T(k_C) - T(k_C - 1) - h\dot{T}(k_C - 1) \tag{8.39}$$
$$e_6(k_C) = T_M(k_C) - T_M(k_C - 1) - h\dot{T}_M(k_C - 1) \tag{8.40}$$
$$e_7(k_C) = T_J(k_C) - T_J(k_C - 1) - h\dot{T}_J(k_C - 1) \tag{8.41}$$
$$e_8(k_C) = T(k_C) - T(k_C - 1) - h\dot{\overline{T}}(k_C - 1) \tag{8.42}$$
$$e_9(k_C) = T_M(k_C) - T_M(k_C - 1) - h\dot{\overline{T}}_M(k_C - 1) \tag{8.43}$$
$$e_{10}(k_C) = T_J(k_C) - T_J(k_C - 1) - h\dot{\overline{T}}_J(k_C - 1) \tag{8.44}$$
$$e_{11}(k_C) = \dot{T}(k_C) - \dot{\overline{T}}(k_C) \tag{8.45}$$
$$e_{12}(k_C) = \dot{T}_M(k_C) - \dot{\overline{T}}_M(k_C) \tag{8.46}$$
$$e_{13}(k_C) = \dot{T}_J(k_C) - \dot{\overline{T}}_J(k_C) \tag{8.47}$$

where $\dot{T}_J$ is the time derivative for the jacket calculated from the model with nominal parameters and measurements, and $\dot{\overline{T}}_J$ is the time derivatives for the jacket from the model for a nominal batch, at sample k_C.

Monitoring of the Cooling Phase

Here the DMP and DMA methods are implemented for the cooling phase of the simulated batch process. A total number of 19 different residuals

for cooling phase have been tested. Not all of these could be used in the DMP/DMA implementation because the residuals did not contain enough information about the different faults or the noise level was too large. To select which residuals that could be used simulations have been performed with different levels of measurement noise.

In the end seven residuals were selected to be incorporated in the DMP/DMA implementation for the cooling phase. These are the residuals found in Eq. 8.27, 8.28, 8.29, 8.43, 8.44, 8.45, and the following

$$e_{14}(k_C) = Q_M(k_C) - \overline{Q}_M(k_C), x \tag{8.48}$$

where Q_M is the heat transferred to the wall calculated form the model and measurements, and $\overline{Q}_M$ is the same for a nominal batch. A list of the residuals can be seen in Table 8.1.

The choice of these residuals is based on simulations of faults using the reaction model to generate fault signatures. The faults that have been considered are a deviation in the initial concentration of A , C_{A0}, a ramp increase in the activation energy of reaction number one and two, E_1 and E_2, initial deviations of E_1 and E_2, and a ramp decrease of the inner heat transfer coefficient, h_i. This gives a total of six different faults, see Table 8.2.

In Fig. 8.2 the fault signature of a ramp in E_1 from 90 to 125 minutes is shown. The temperature in the reactor is controlled and does not deviate much from its set point. To achieve this the amount of heat removed from the reactor must be lowered, which can be seen in the decrease of the position of the cooling valve. The temperature in the jacket will then increase and so will the temperature in the wall. Here one can draw the conclusion that the temperature measurements in the wall and in the jacket can be used for the detection of this fault. The position of the

Table 8.1 Residuals used for monitoring of the cooling phase.

Residual	Variable	
1	Reactor temperature T	e_1
2	Wall temperature T_M	e_2
3	Jacket temperature T_J	e_3
4	Estimated time derivative of T_M	e_9
5	Time derivative of T	e_{11}
6	Time derivative of T_M	e_{12}
7	Heat transferred to wall Q_M	e_{14}

Table 8.2 Definition of faults.

Fault	Description
1	Ramp in E_1
2	Ramp in E_2
3	Ramp in h_i
4	Initial increase in E_1
5	Initial increase in E_2
6	Increase of C_{A0}

cooling valve is also a candidate. This is, however, not possible when measurement noise is added, as can be seen in Fig. 8.3. This is due to the aggressive tuning of the controller. The two temperature measurements are also not as good indicators as before but they are still possible to use in the detection.

Tolerances for the Cooling Phase One problem when choosing the tolerances is that the level of noise in the different signals may destroy the residual information. For example, an aggressive feed-back controller will generate a lot of random noise in the control signal and in the flow in the jacket, which first affects the temperature in the jacket and then the temperature in the wall, and so on. This makes it harder to use the residuals associated with the states first affected by the feed-back if it is possible at all. Especially residuals using an approximation of a time derivative calculated by a difference approximation is sensitive to noise. Only when the residual is very large this can be used. Since the tolerances must be set to avoid too many false alarms residuals of this type may not always be applicable.

During the evaluation of the DMP methodology the approach of using only one tolerance per residual, i.e., τ_m, and then calculating sensitivities according to Eq. 8.5 did not give successful results. Instead tolerances associated with both residuals and fault, i.e., τ_{mn}, were used. The selection of the tolerances are critical for the performance of the fault detection of this process. The fault signatures described above can be used to determine the tolerances. From Fig. 8.3 it can be determined that the tolerance for when the ramp has increased E_1 with 1% for the two temperature measurements should be chosen to approximately 1^oF.

For a batch process the tolerances and/or sensitivities may vary during the duration of the batch. For example, a deviation in E_2 will have a very small influence on the process at the beginning of the batch since the

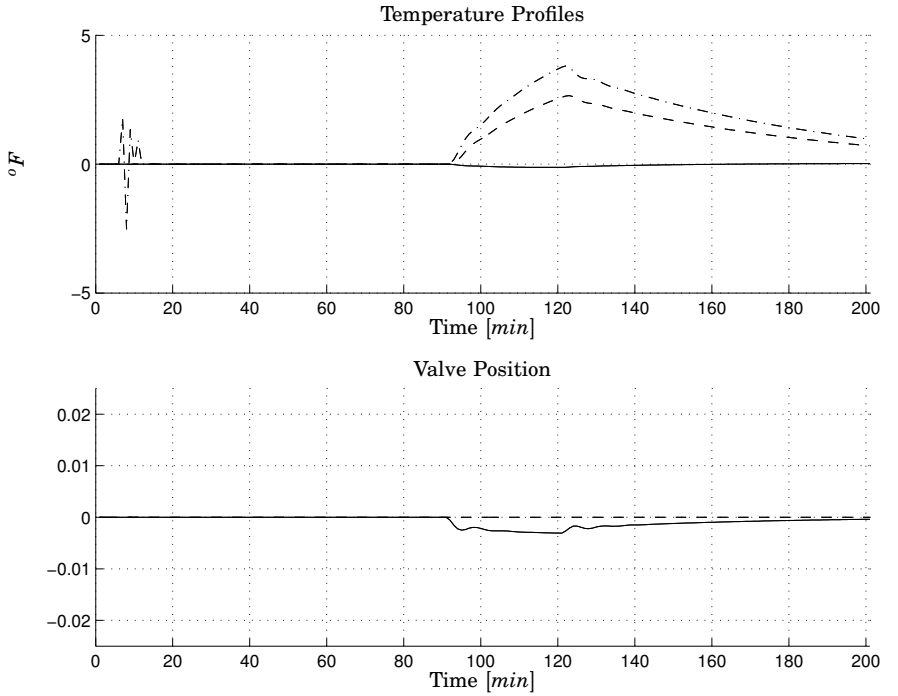

Figure 8.2 Influence of positive ramp in E_1 from 90 to 125 minutes without measurement noise. The upper plot shows the deviations from the nominal batch in the temperature profiles: inner reactor temperature T (solid), wall temperature T_M (dashed), and jacket temperature T_J (dash-dotted). The lower plot shows the heating (dashed) and cooling valve (solid) positions. The deviations in T_J at around 10 minutes is because of the two phase steam flow.

concentration of B is very small, but then it increases as the reaction goes on. One way of choosing how the tolerances should vary with time is to simulate the same fault at different times over the batch duration and then interpolate between these time instances. In this implementation the tolerances have only been calculated for a few specific time spans.

Probably the most correct way of calculating the sensitivity to parameter faults is by using the sensitivity function defined in [Khalil, 2002], where the sensitivity is calculated using a dynamic equation based on the process equations. This sensitivity function based on the measured signals and the nominal measurement trajectories gives approximations of the partial derivatives of the states with respect to the parameters. Another approach for adaptive threshold generation for DMP can be found

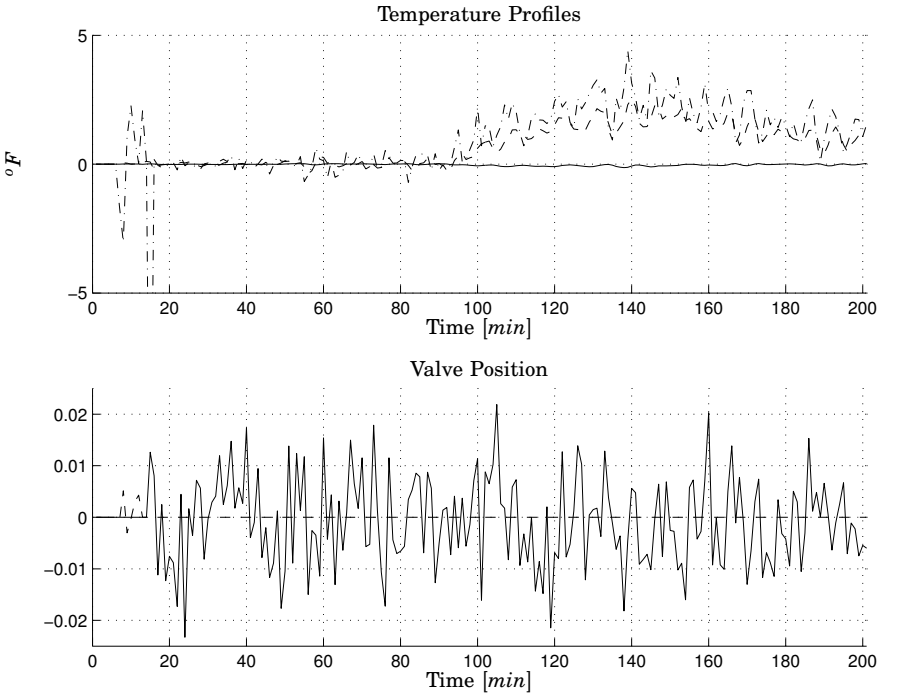

Figure 8.3 Influence of positive ramp in E_1 from 90 to 125 minutes with measurement noise. The upper plot shows the deviations from the nominal batch in the temperature profiles: inner reactor temperature T (solid), wall temperature T_M (dashed), and jacket temperature T_J (dash-dotted). The lower plot shows the heating (dashed) and cooling valve (solid) positions. The deviations in T_J from 10 to 20 minutes is because of the two phase steam flow.

in [Puig *et al.*, 1997], where DMP is applied to a continuous process.

Results for the Cooling Phase After determining the tolerances the implementation was tested on the faults. Two of the faults are described in more detail here. The first is the deviation in C_{A0} (Fault 6) and the second is a ramp increase of the activation energy of the first reaction E_1 (Fault 1).

DMP The first index calculated is the satisfaction value. As mentioned before the approach from DMA with a specific tolerance τ_{mn} for the mth residual and nth fault is used. The magnitude of the sensitivities S_{mn} are chosen to be equal to unity.

The satisfaction values calculated from Eq. 8.10 for the batch with a deviation in C_{A0}, Fault 6, can be found in Fig. 8.4. The small plots show the satisfaction value for a specific residual and fault from 20 to 200 minutes of the total batch run, i.e., the cooling phase. An empty plot in the figure means that the residual is not sensitive to this fault.

The residuals for Fault 6, the column furthest to the right in Fig. 8.4, all show a satisfaction value close to 1 for the first part of the plot indicating a high value of C_{A0}. The reason that the satisfaction value goes down after a certain time is that a constant tolerance is used, which is only valid for the first part. The reason for this is that it is not very likely that C_{A0} will be a probable fault after the initial part of the batch. In fact the monitoring of this fault should probably be excluded after the first part of the batch not to give any false alarms.

In Fig. 8.4 it can also be seen that all the satisfaction values for Fault 1, 2, and 5 have a value close to -1. Fault 1 and 4 are related to each other since they both concern a deviation in E_1. Fault 1's tolerance is calibrated for a change in E_1 occurring somewhere in the middle of the batch, i.e., around 100 minutes and this is the reason this fault will get a large negative failure likelihood at the beginning of the batch, see Fig. 8.6, while the tolerance of Fault 4 is calculated for an initial deviation of E_1 and does not get as large negative failure likelihood. This implies that Fault 1 should not be monitored until the time span for which the tolerance has been calculated is reached. In the same way Fault 2 and 5 are related to each other. Fault 2 should be ignored until the middle of the batch. Fault 5 is the initial deviation in E_2 and the failure likelihood is not able to distinguish between this fault and Fault 6, see Fig. 8.6 where the failure likelihood is plotted.

The second fault considered is a ramp in E_1 occurring after 90 minutes, i.e, Fault 1. In Fig. 8.5 it can be seen that all the residuals concerning Fault 1 and 2 have very similar satisfaction values. This is not very surprising since the two fault influence the process in a very similar way and at this time of the batch the concentrations of A and B are close to each other. This means that the two faults cannot be distinguished from one another using DMP, see Fig. 8.7 where the failure likelihood is plotted.

DMA In DMA the satisfaction values for the residuals that are independent of a fault are also calculated using Eq. 8.11 and then the degree of fault is determined from Eq. 8.14. In Fig. 8.8 this is shown for a batch with deviation in C_{A0}. Here the degree of fault of Fault 6 is equal to 1 during the beginning of the batch. Fault 1 and 2 show a large negative value but again the tolerances for these faults are not calculated for this time interval. Fault 4 and 5 also show large negative values. Here it should be remembered that this plot should be used together with the plot of the

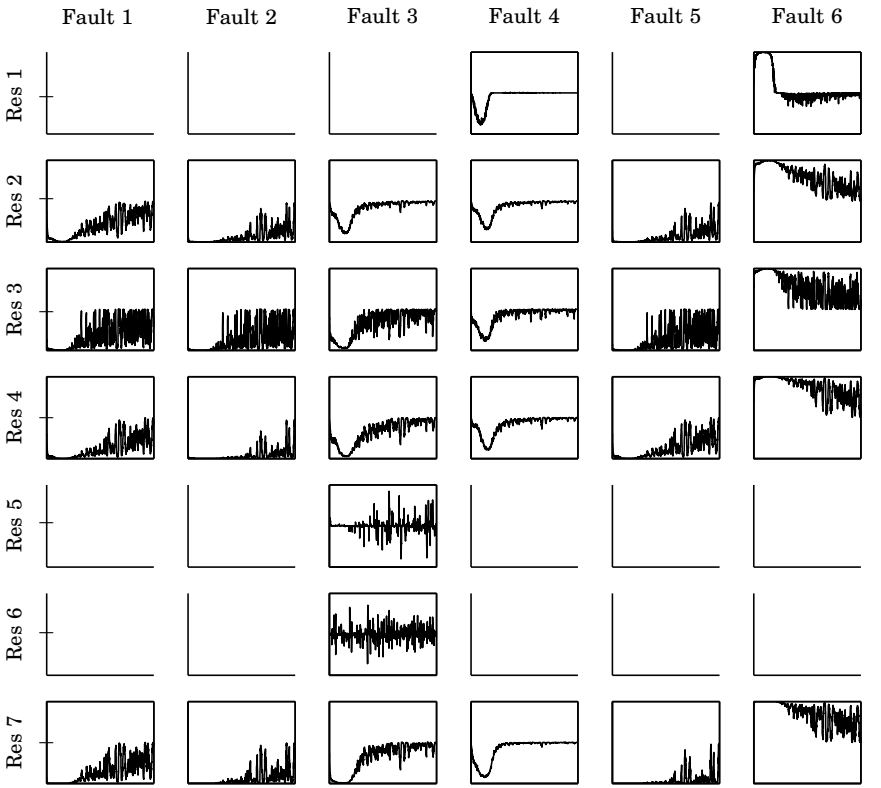

Figure 8.4 Satisfaction values in DMP for a batch with 3% high C_{A0} (Fault 6). The x-axis of each sub-plot is the duration of the Cooling phase, i.e., from 20 to 200 minutes, and the y-axis is the satisfaction value ranging from -1 to $+1$.

consistency factor, Fig. 8.10. In this figure it can be seen that only Fault 6 have a consistency factor close to 1 and should therefore be identified as the cause of this upset.

For the monitoring of a batch with a ramp in E_1 from 90 to 125 minutes, i.e., Fault 1, the plot of the degree of fault can be found in Fig. 8.9, where Fault 1 and 2 show positive values close to 1 from around 100 minutes. The consistency factors for both of the faults are close to 1 and therefore the faults cannot be distinguished from each other using DMA, either, see Fig. 8.11. Both Fault 5 and 6 show large positive and negative values respectively for the degree of fault in Fig. 8.11, but these are con-

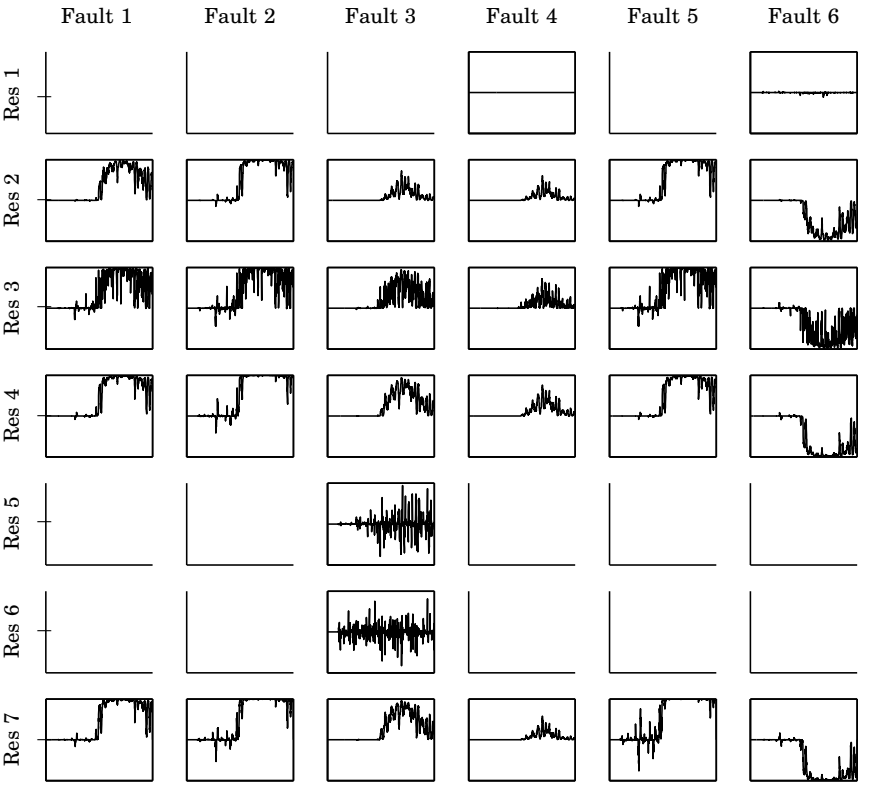

Figure 8.5 Satisfaction values in DMP for a batch with 1% ramp E_1 from 90 to 125 minutes (Fault 1). The x-axis of each sub-plot is the duration of the Cooling phase, i.e., from 20 to 200 minutes, and the y-axis is the satisfaction value ranging from -1 to $+1$

cerned with initial deviations and should not be considered as alternatives at this point in the batch.

It should be noted that in the plots of the degree of fault the flickering mentioned above can be seen. Especially in Fig. 8.9 where the batch is normal for the first part, but for Fault 2 and 5 the degree of fault jump between plus and minus 0.5 at several time instances.

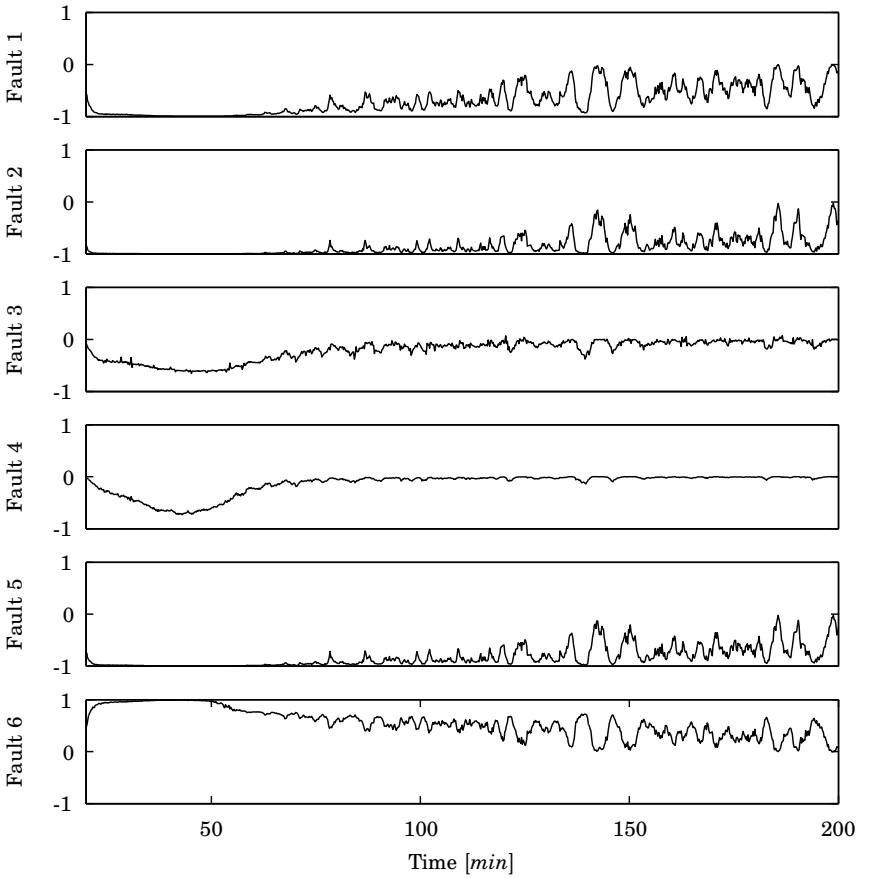

Figure 8.6 Failure likelihood in DMP for a batch with 3% high C_{A0} (Fault 6).

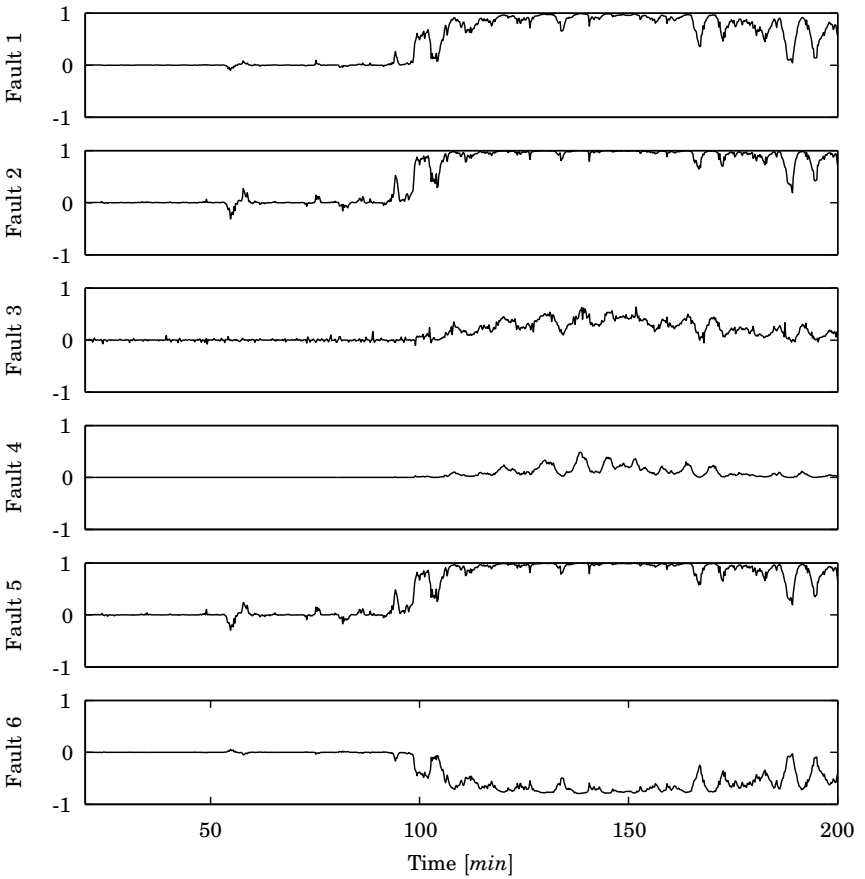

Figure 8.7 Failure likelihood in DMP for a batch with 1% ramp E_1 from 90 to 125 minutes (Fault 1).

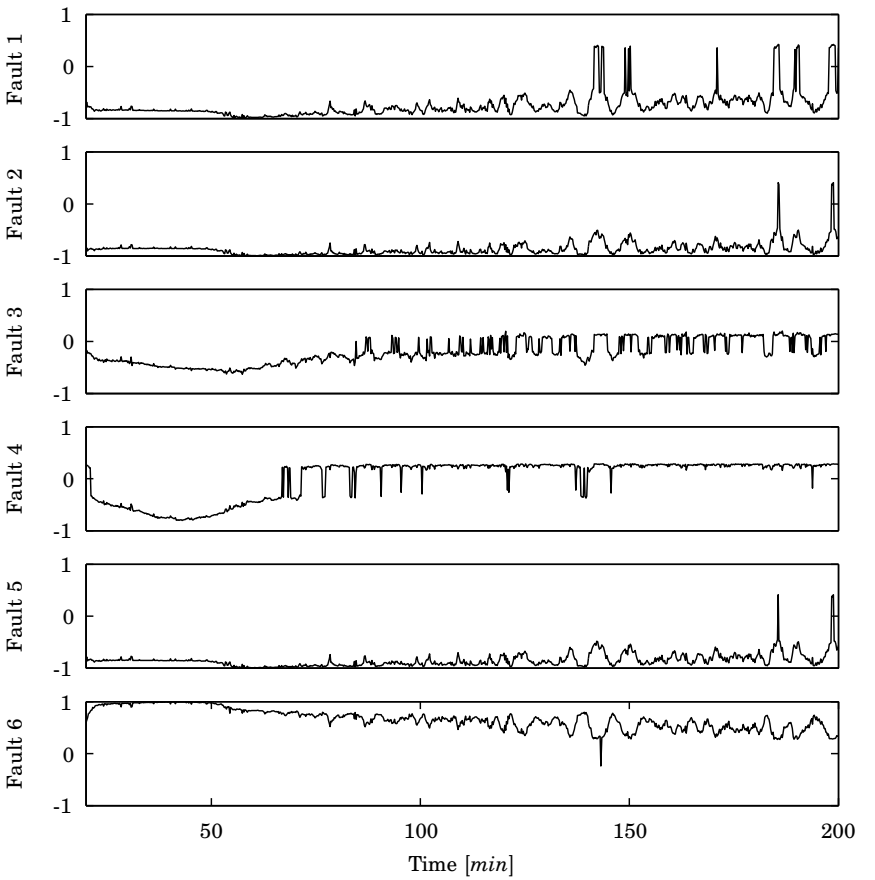

Figure 8.8 Degree of fault in DMA for a batch with 3% high C_{A0} (Fault 6).

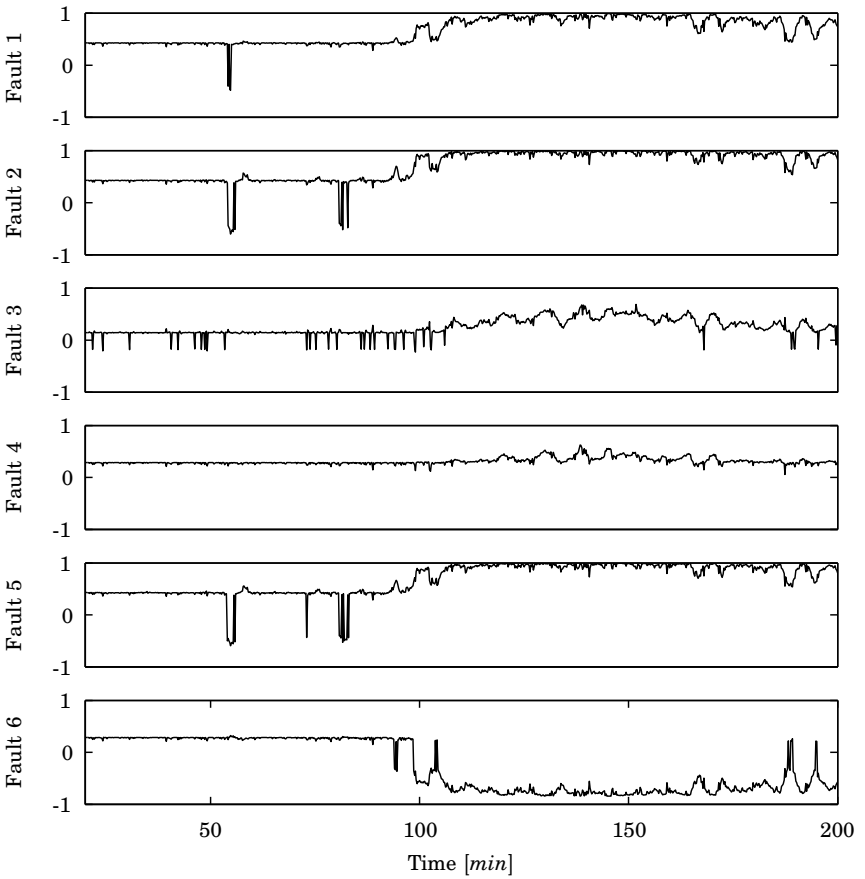

Figure 8.9 Degree of fault in DMA for a batch with 1% ramp E_1 from 90 to 125 minutes (Fault 1).

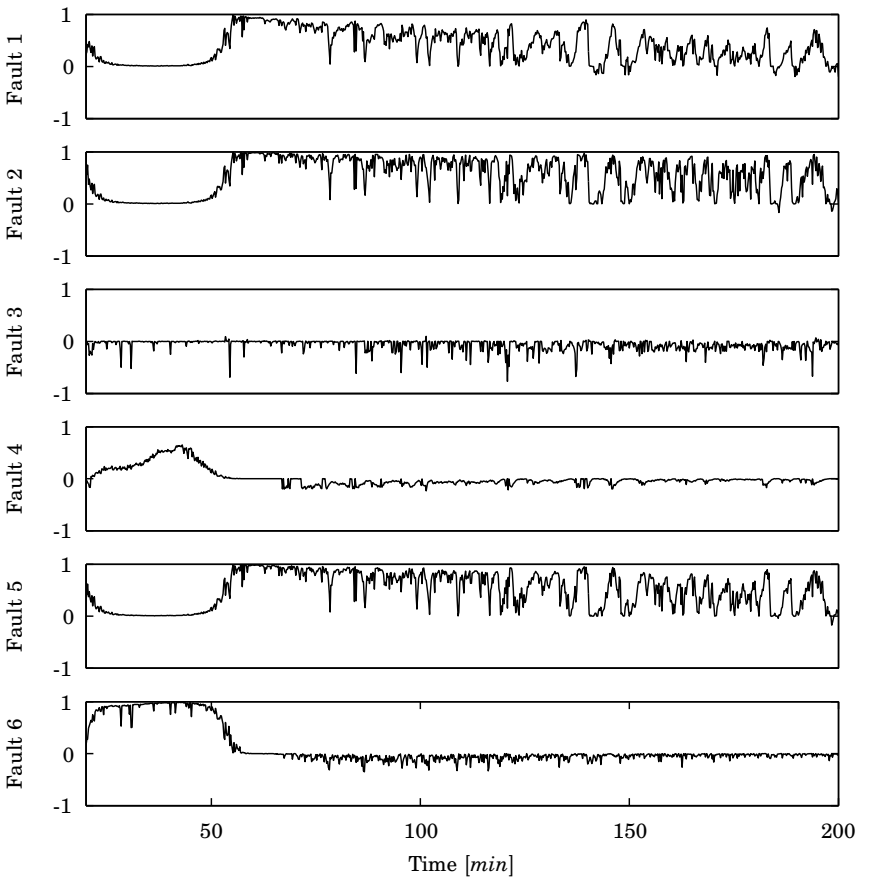

Figure 8.10 Consistency factor in DMA for a batch with 3% high C_{A0} (Fault 6).

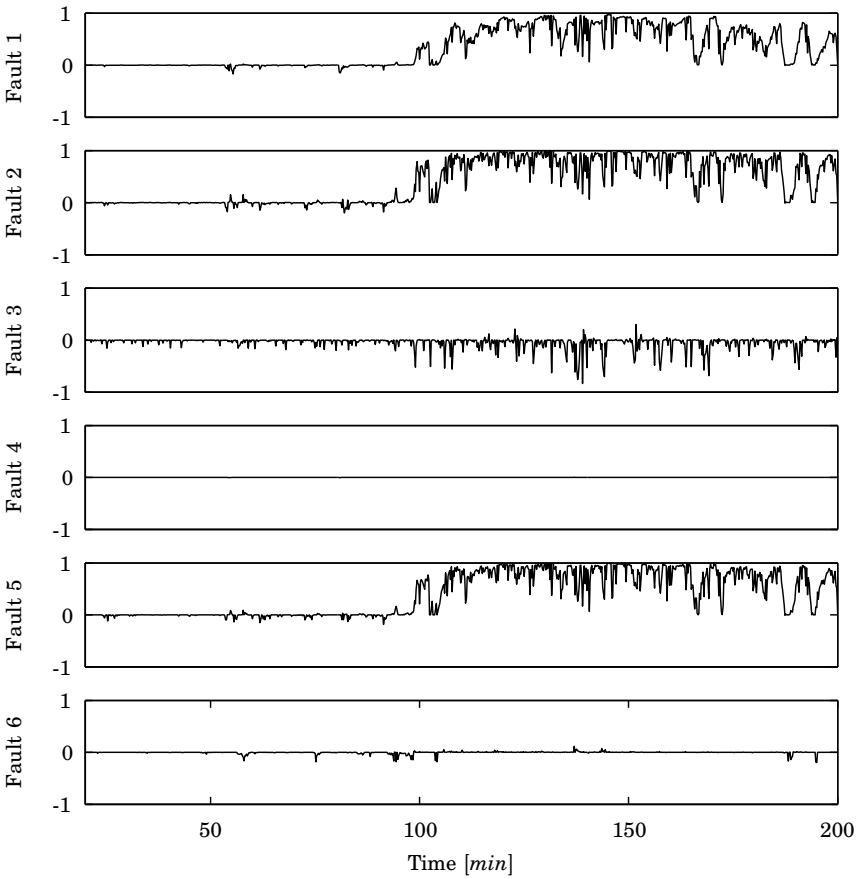

Figure 8.11 Consistency factor in DMA for a batch with 1% ramp E_1 from 90 to 125 minutes (Fault 1).

8.5 Summary

In this chapter the two methods DMP and DMA have been discussed and applied to a batch process. It is shown under the assumption that every batch is running under the same nominal conditions that the methods can detect single faults, but not all faults can be isolated from each other. The DMP method signals all faults that are possible and further diagnosis is needed to isolate a specific fault. The adding of supportability is a step in this direction. In DMA this is comparable to the adding of the satisfaction value for residuals that are independent of the fault. The consistency factor in DMA also improves the isolation properties compared to DMP.

The other faults in Table 8.2 have also been tested using DMP and DMA. Fault 3, i.e., a change in the inner heat transfer coefficient, h_i, is the easiest to detect and isolate, and both the methods behave similarly. The fault that is hardest to detect is Fault 5, an initial deviation in E_2. This is because at the beginning of the cooling phase there is little B in the reactor and the influence from the fault is small. This leads to that a small tolerance needs to be used, which leads to that this fault is often signaled as a probable fault even though it is not correct.

Since there will always be differences between batch runs such as initial conditions, parameter changes, disturbances, and measurement noise a successful batch where the end product is within the given specifications does not need to exactly follow the nominal trajectory. If a data base of historically successful batches are available upper and lower limits of the residuals that gives the specified end product can be calculated. These batch to batch differences affect the performance of the methods and should be taken into consideration when determining the limits, τ_{mn}. Since a batch process is time variant the limits in the methods need to be determined for different time intervals, requiring a lot of work effort. It has also been shown that the methods are sensitive to noise and that the residuals need to be chosen with care to achieve good performance.

In both DMP and DMA the residuals are weighted and transformed using a non-linear function, the sigmoidal function in Eq. 8.4 and 8.10. The use of these functions will hide information about the magnitude of the satisfaction. There is no difference between a residual which has the magnitude of 4 or 5 τ. The satisfaction for both is equal to unity. The developers of the methods point out that at this level of the diagnosis all faults should be shown as possible and that further diagnosis should be performed to conclude which fault has occurred.

9

Comparing Multivariate Statistical Methods for Batch Process Monitoring

9.1 Introduction

In this chapter a comparison between the methods multi-way principal component analysis, MPCA, using different ways of unfolding the data, batch dynamic PCA, BDPCA, multi model MPCA, and moving window PCA described in Chapter 6 is performed. The batch process described in Chapter 7 is used as a benchmark process. In real batch processes the durations of the batches are often different. A simple solution to deal with this is to use another monotonic maturity variable than time to describe the development of the batch from start to end. Another approach is to use dynamic time warping, a method from speech recognition, to align batches with different duration, see [Kassidas *et al.*, 1998]. Here, to simplify the comparison, an equal batch length for all batches is used. The simulation time for the batches was set to 200 minutes, where the yield of the product B is considered to be optimal for a nominal batch. For the monitoring a sampling time of 1 minute was used, which results in that the number of samples $K = 201$. This is slower than the controller sampling time, which is $12s$.

To build models using the multivariate statistical methods seventy successful or normal batches were used, i.e., $I = 70$. The batches were simulated using Monte-Carlo simulation. The parameters that were varied for the normal batches were the initial concentration, C_{A0}, the heat transfer coefficient inside the reactor, h_i, and the activation energies for

Table 9.1 Standard deviation in the parameters.

Parameter	Deviation
C_{A0}	0.5%
h_i	0.5%
E_1	0.1%
E_2	0.1%

the two reactions, E_1 and E_2. The parameters were assumed to be normally distributed and the standard deviations for the deviation from the nominal values for the parameters are found in Table 9.1.

In the monitoring four variables were used, $J = 4$, the temperature in the reactor, T, the temperature in the jacket, T_J, the temperature in the reactor wall, T_M, and the control signal to heating/cooling valve. Measurement noise was added to the temperature measurements with a standard deviation of 0.1^oF. This leads to that $\underline{X} \in \mathbb{R}^{70\times4\times201}$.

Two faults have been selected to compare the methods. The faults are

- Upset in initial concentration, C_{A0}, by 3%
- Ramp up to 1% higher activation energy for reaction one, E_1, from time 90 to 145 minutes

These two faults are the same as Fault 1 and 4 in Table 8.2 and are considered to have a large impact on the final product. In all of the plots where control limits are included, the 95% limit is plotted using a dashed-dotted line and the 99% limit is plotted using a dashed line.

9.2 MPCA

The two different ways of unfolding the tensor $\underline{X}$ to the matrices $X^{I\times KJ}$ and $X^{IK\times J}$, and then applying PCA are compared in this section.

Batch-Wise Unfolding

This is the method where $\underline{X}$ is unfolded to $X^{I\times KJ} = X^{70\times804}$ used in [Nomikos and MacGregor, 1994; Nomikos and MacGregor, 1995b; Nomikos and MacGregor, 1995a]. The data is mean centered and scaled over the columns to remove the non-linear trajectories over time. In this implementation the projection to model plane method [Arteaga and Ferrer, 2002] has been used to fill in future measurements.

For the projection to model plane method to be able to give good estimations of the future measurements data collected from the new batch up to 20 samples was needed. Before this the control limits become very large, probably also due to that the change between the heating and the cooling phase occurs a few samples before this instance as well. In other implementations in the literature it have been reported that 10% of the data from the whole batch usually gives a good prediction of the future measurements. This is the reason why the monitoring and the control limits do not start until the 20th sample in the figures with the results from the method

A model with three principal components was selected based on the singular values. The control limits for the scores, SPE, and T^2 were calculated according to Chapter 6. In Fig. 9.1 a batch run with nominal parameters is shown. The scores have values close to zero, which classifies the batch as being close to the 'mean batch' of the batches used for calculating the model. The SPE shows a low value except for some high peaks. This leads to the conclusion that the batch is correctly classified as a normal batch.

In Fig. 9.2 a batch with a 3% positive initial deviation in C_{A0} is shown. This upset is detected soon after the first twenty samples have been collected when the score of the first principal component goes outside its control limits. The batch stays outside the control limits for the rest of the batch duration and the batch is classified as being abnormal. The SPE is not sensitive to this fault, which means that the model still is able to describe the batch but its distance to the 'mean batch' is large.

In Fig. 9.3 the monitoring of a batch where the E_1 is ramped up 1% from 90 to 145 minutes is shown. The upset is detected in the SPE at around 110 minutes when the value starts to ramp up. The second principal component goes outside its control limit at 145 minutes. After 145 minutes the SPE starts to decrease to finally end up inside its limit at the end of the batch. This can be explained by remembering that the batch was normal until 90 minutes. This means that after, e.g., 100 minutes data from 90 minutes of normal operation and data from 10 minutes where the process is starting to drift is used for filling in the future. This cannot be described by the model based on the three principal components and therefore the SPE will show a high value. As the batch continues more data from the faulty process is used and the effects of the fault gradually be show up more and more in the T^2 since more and more of the batch can be described by the model.

It has been shown that the method is able to correctly classify the two faulty batches as faulty and that a normal batch is clearly shown as such.

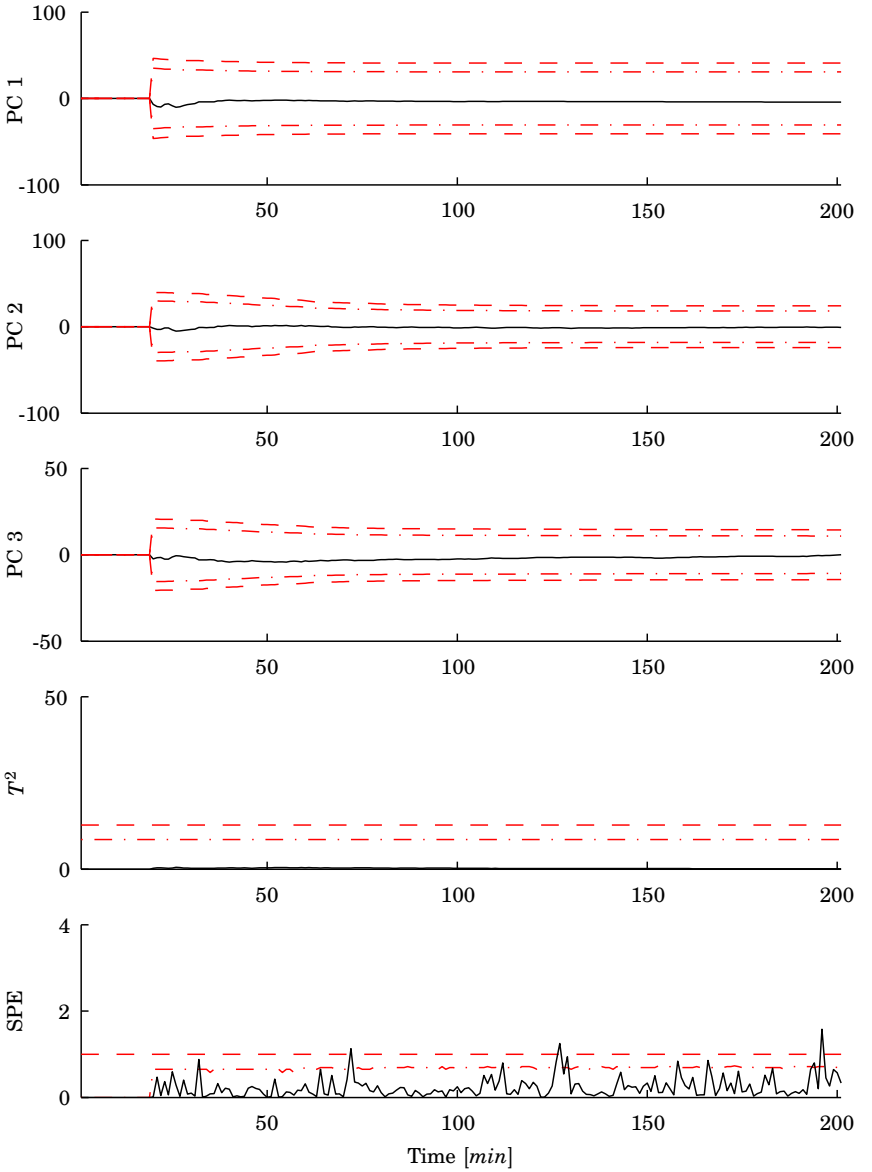

Figure 9.1 Monitoring of a batch with nominal parameters using MPCA and batch-wise unfolding.

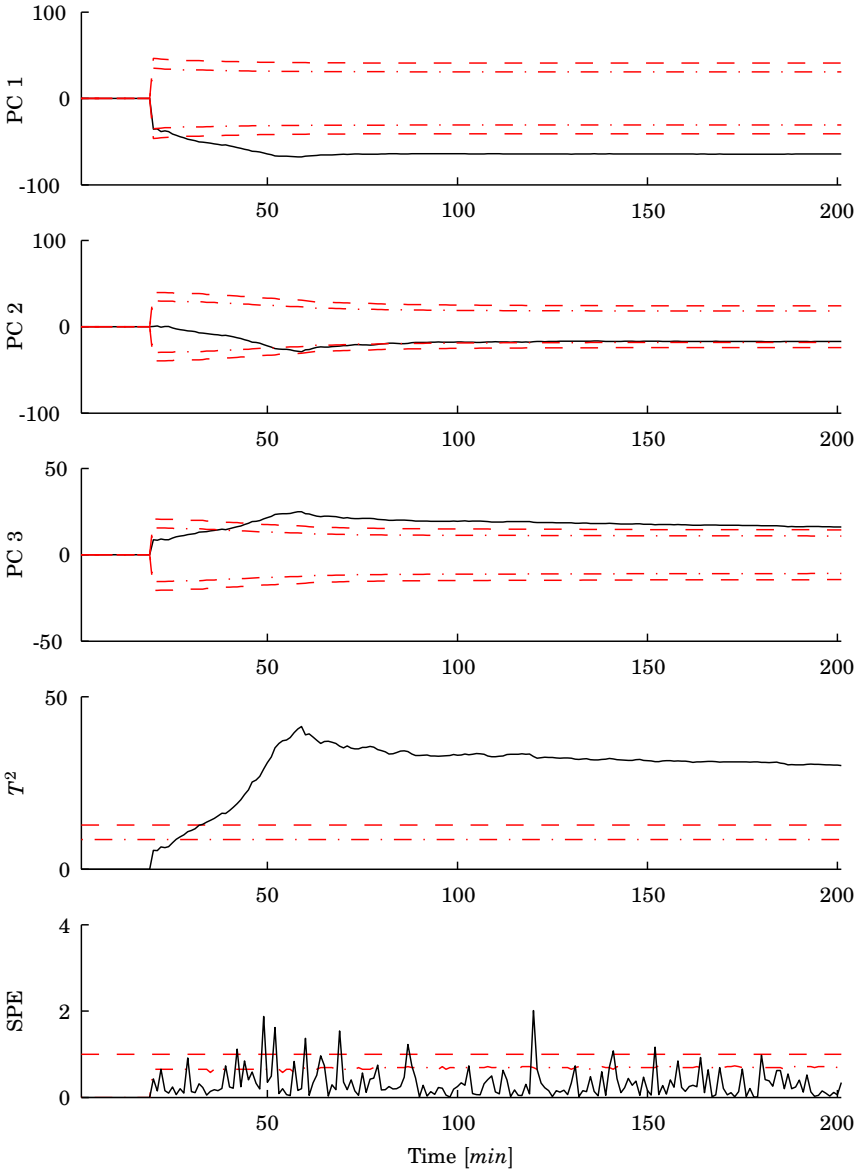

Figure 9.2 Monitoring of a batch with 3% high C_{A0} using MPCA and batch-wise unfolding.

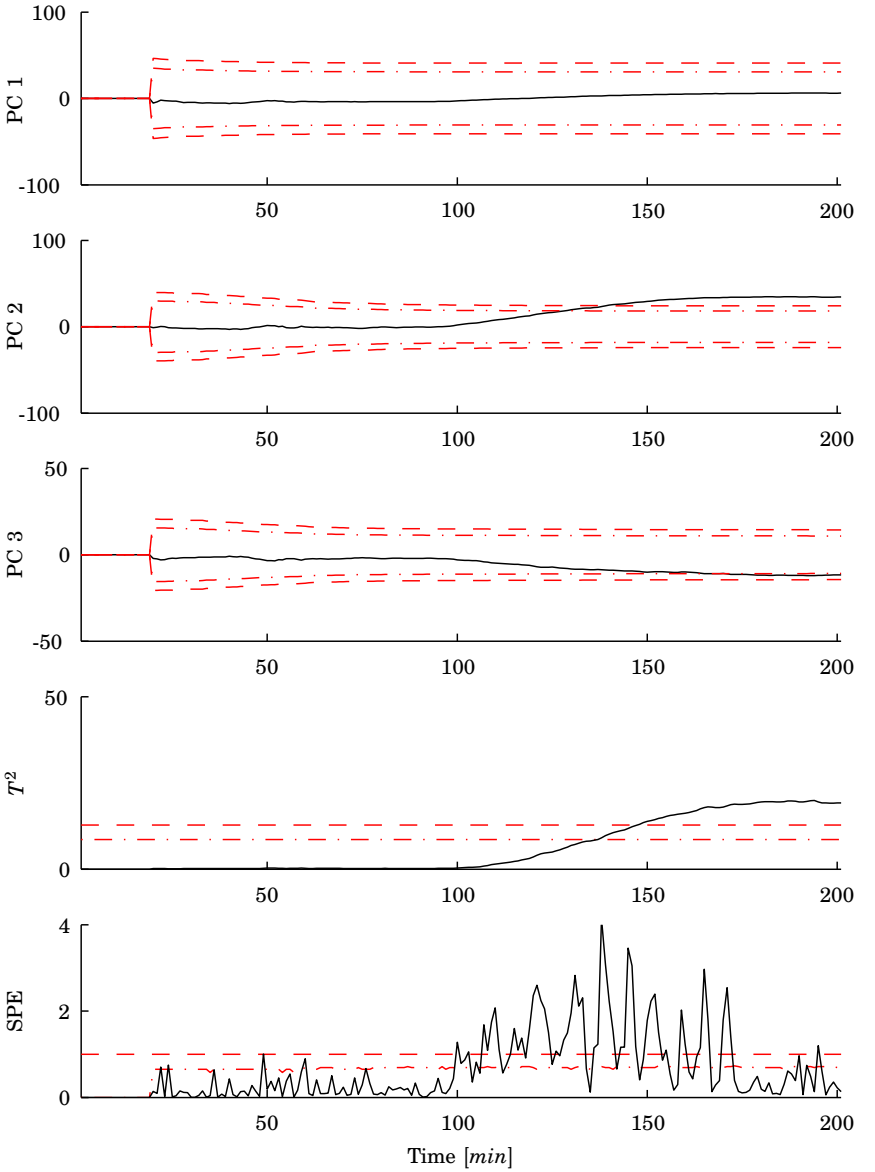

Figure 9.3 Monitoring of a batch with a 1% ramp in E_1 from 90 to 145 minutes using MPCA and batch-wise unfolding.

Variable-Wise Unfolding

The second way of MPCA is to unfold $\underline{X}$ to the matrix $X^{IK\times J} = X^{14070\times 4}$. Two different version of this method is implemented here. The first is the version described in [Wold *et al.*, 1998] where the non-linear dependencies are left in the data since only the grand mean is subtracted from the data. The second version is to first remove the non-linearities by mean centering and scaling in the same ways as for the $X^{I\times KJ}$ unfolding and then unfold the data to $X^{IK\times J}$ [Chen *et al.*, 2002].

First Version In the first version a model with two principal components was selected based on the singular values. The maximum number of components in this method is $J = 4$. The control limits for the scores, SPE, and T^2 were calculated according to Chapter 6. In Fig. 9.4 a batch run with nominal parameters is shown. As can be seen this way of displaying the score values is not very good. Since the trajectories over time are not subtracted it is very hard to see the control limits or the values of the batch. It is not possible for an operator to see if the batch is inside the control limits or not in the score plots. Instead in the rest of the score plots in this section the mean of the score trajectories will be subtracted for better display, i.e., in Fig. 9.5 and 9.6. Since the scores and thus T^2 are based on only the latest measurement, measurement noise have a great effect. This is the reason for the spikes outside the control limits in the T^2 plot even though the batch monitored is a nominal batch.

In Fig. 9.5 the monitoring of a batch with a 3% positive initial deviation in C_{A0} is shown. The upset is detected after approximately 25 minutes in the T^2 plot. As can be seen the score plots are now more operator friendly. To be able to zoom in on the scores the high tops at around 20 minutes, which are due to the switching between the heating and the cooling the, have been cut off in the plot. The top value of the peaks is almost two which is ten times larger than the scale in the plot. A problem with this method can be noticed. One can easily believe that the batch has been brought back to a normal state at around 100 minutes. When using this method if the batch moves outside its limits during only a short period the product might not be on the specifications. This is again because the monitoring at each time instance is based only on the latest measurement.

In Fig. 9.6 the monitoring of a batch where E_1 is ramped up 1% from 90 to 145 minutes is shown. The upset is detected in T^2 at around 120 minutes but the values moves in and out of the region of normal operation, which again can confuse the operator.

Both the need to subtract the mean from the score plots and the way of determining the control limits described in Chapter 6 can be seen as a way to compensate for leaving the non-linear trajectories in the data matrix $X^{IK\times J}$ when calculating the model.

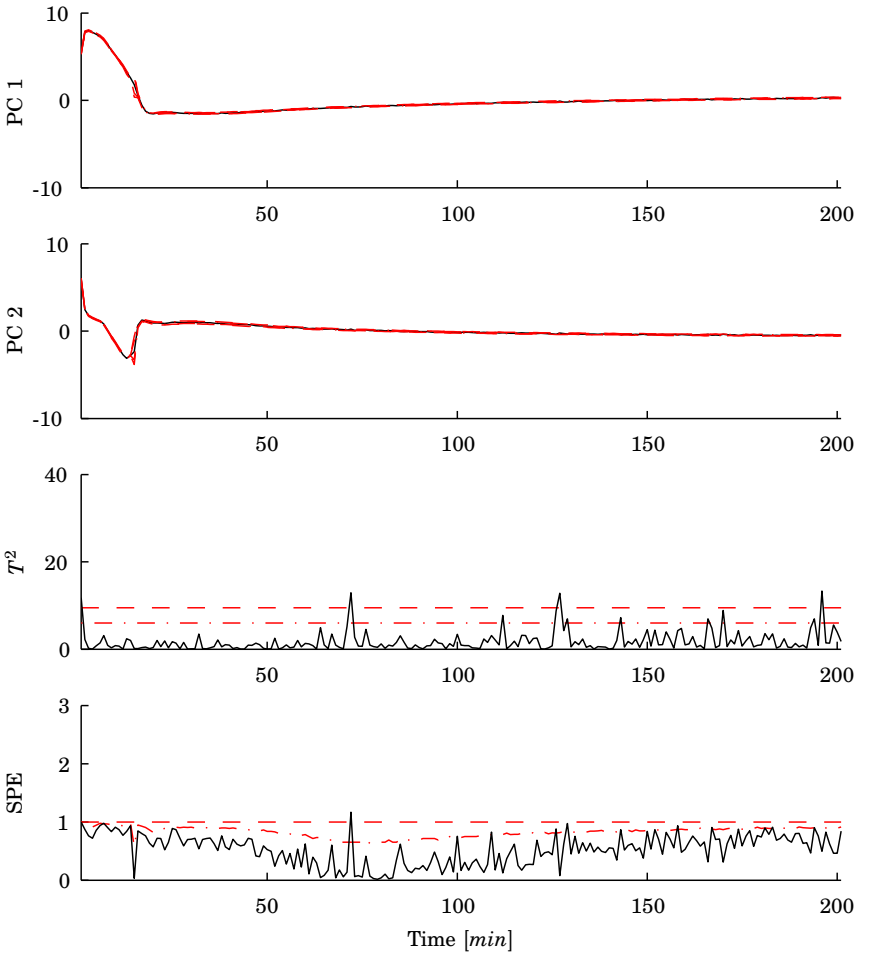

Figure 9.4 Monitoring of a batch with nominal parameters using MPCA and variable-wise unfolding, first version.

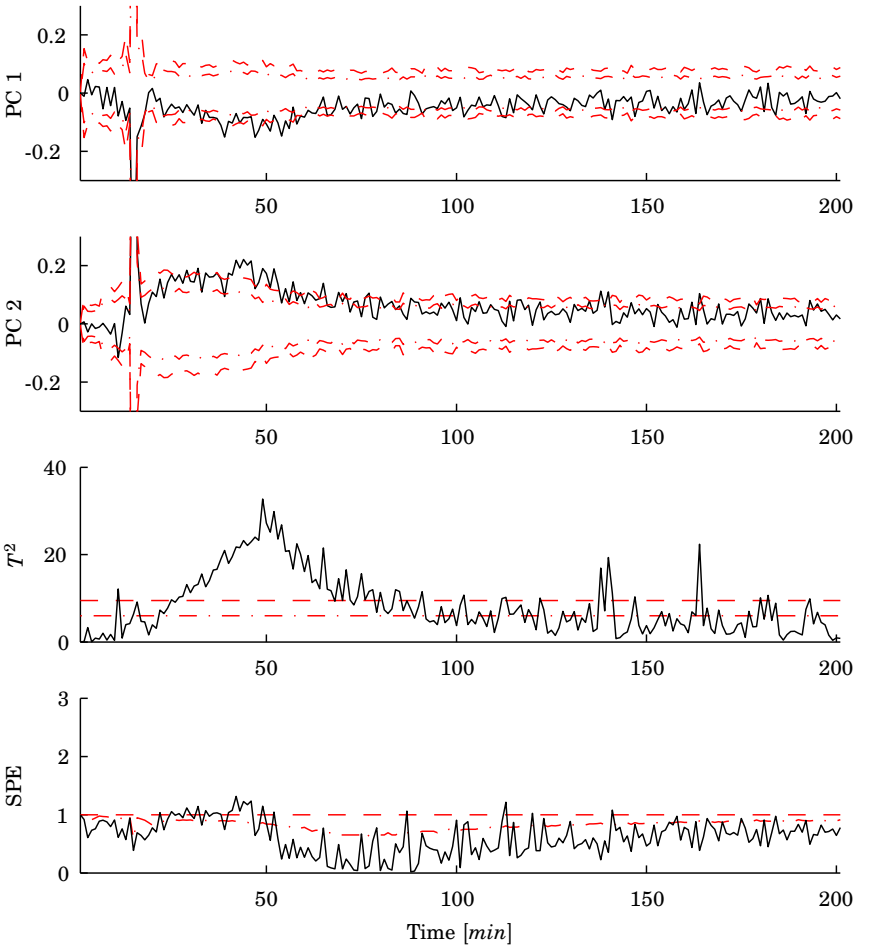

Figure 9.5 Monitoring of a batch with 3% high C_{A0} using MPCA and variable-wise unfolding, first version. The mean at each time have been subtracted in the score plots.

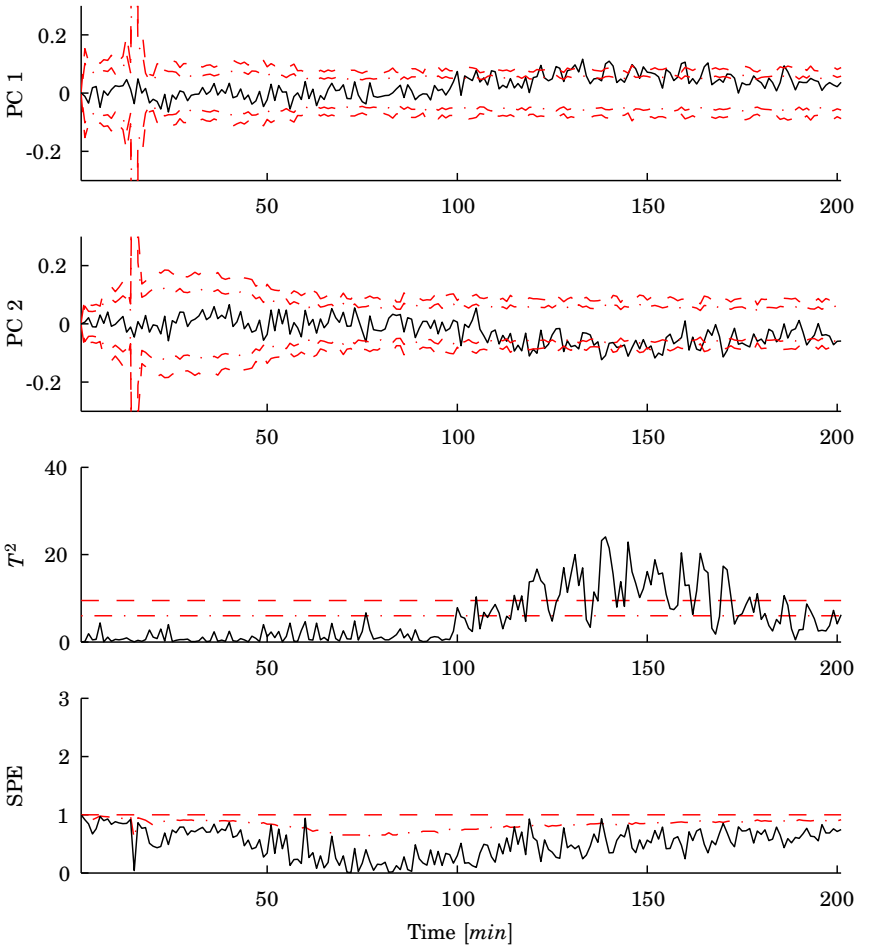

Figure 9.6 Monitoring of a batch with a 1% ramp in E_1 from 90 to 145 minutes using MPCA and variable-wise unfolding, first version. The mean at each time have been subtracted in the score plots.

Second Version The second version model also uses two principal components of the maximum four. The control limits for the scores, SPE, and T^2 were again calculated according to Chapter 6. In Fig. 9.7 a batch run with nominal parameters is shown.

In this version the means of the trajectories do not have to be subtracted since the data have already been mean centered at each time point. The scores and thus the T^2 are still based only on measurements from one time instance and the scores still look noisy, moving around randomly inside the limits.

In Fig. 9.8 the monitoring of a batch with a 3% positive initial deviation in C_{A0} is shown. The upset is detected after approximately 25 minutes in the score plot of the first principal component and in the T^2 plot. After 60 minutes the second score plot changes dynamically from the upper to lower control limit, which would make it harder for an operator to make a decision of what is wrong with the batch. The change is because the process itself changes behavior at this time instance. This is further discussed in the section about moving window PCA. During the period of change the SPE is rising showing that the current batch behavior is not described by the model. The batch is outside the control limits of SPE from around 75 minutes to the end of the batch, while T^2 moves back inside its limits at the end.

In Fig. 9.9 the monitoring of a batch where the E_1 is ramped up 1% from 90 to 145 minutes is shown. The upset is detected in the score plot of the first principal component and in the T^2 at around 110 minutes. At the end, the batch moves closer to the region of normal operation to finally be inside.

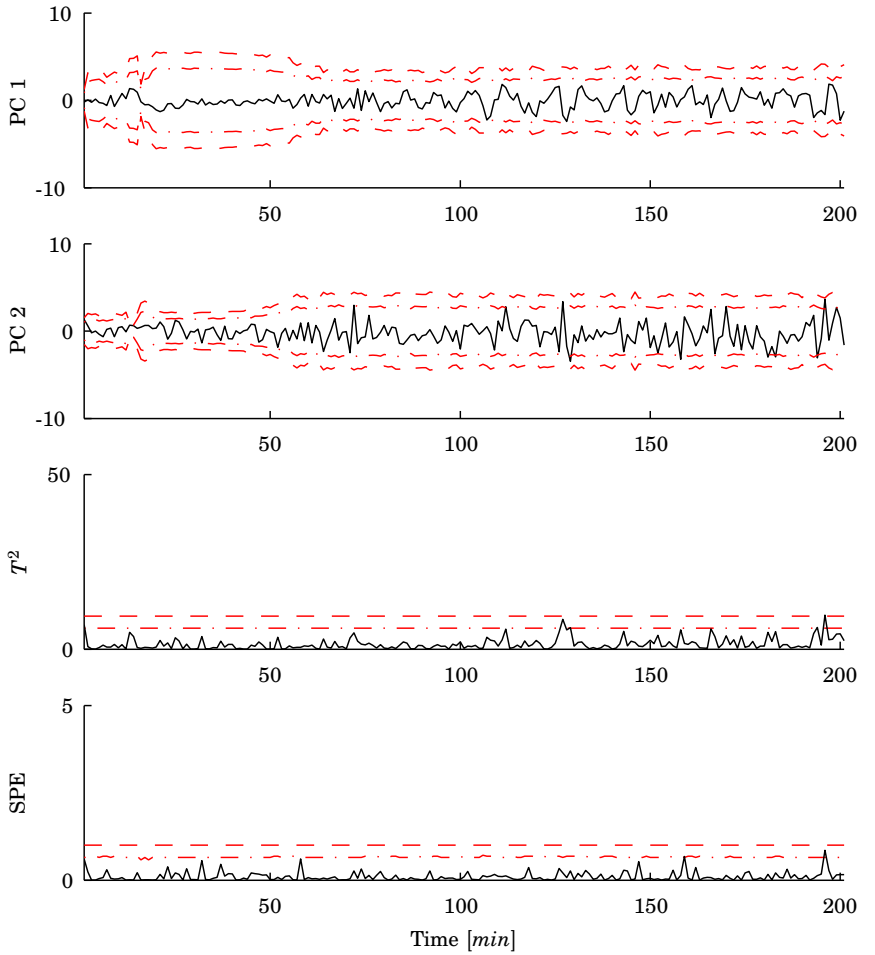

Figure 9.7 Monitoring of a batch with nominal parameters using MPCA and variable-wise unfolding, second version.

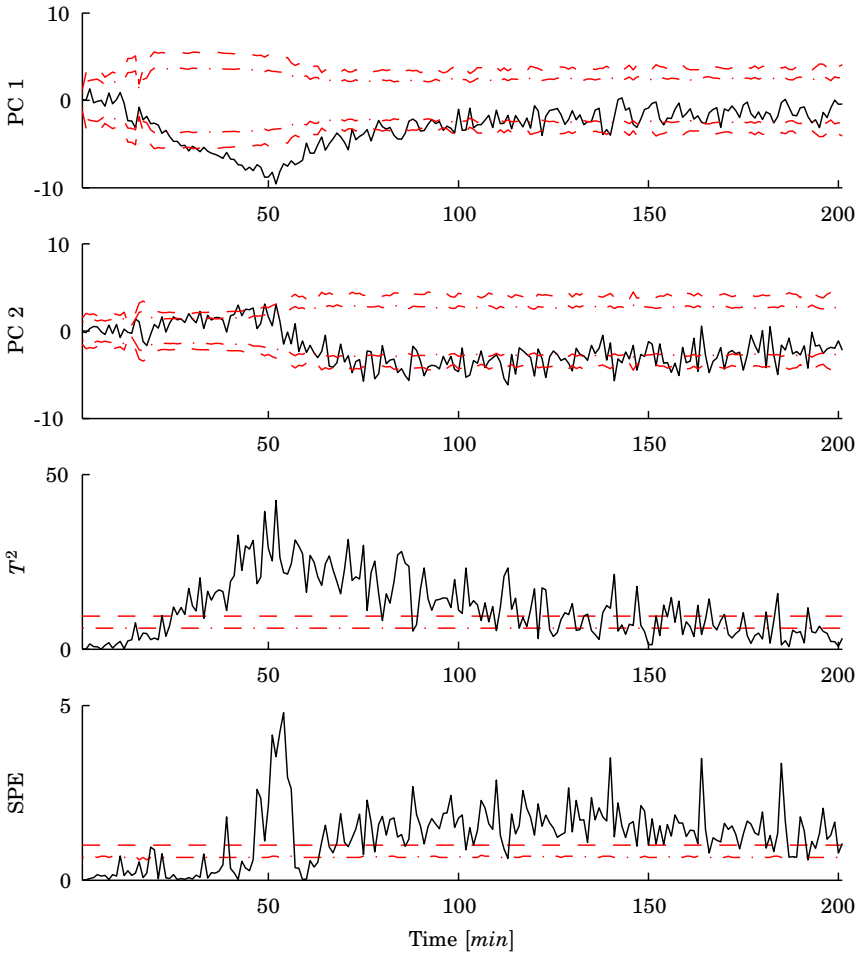

Figure 9.8 Monitoring of a batch with 3% high C_{A0} using MPCA and variable-wise unfolding, second version.

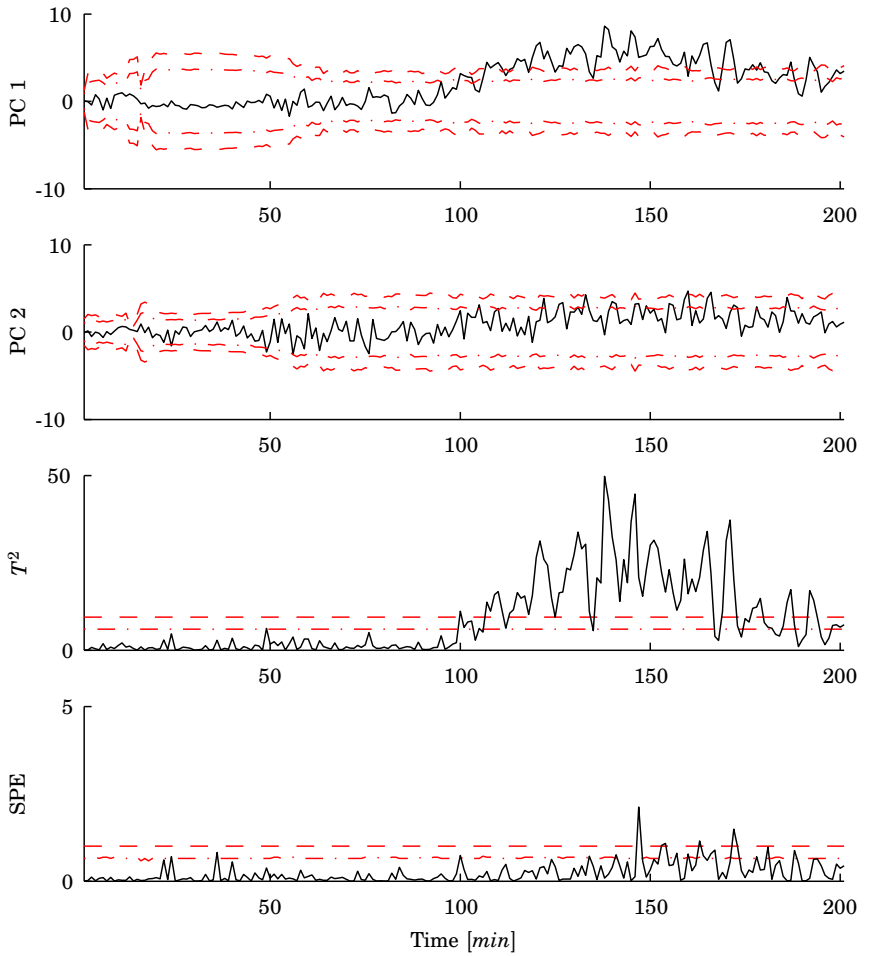

Figure 9.9 Monitoring of a batch with a 1% ramp in E_1 from 90 to 145 minutes using MPCA and variable-wise unfolding, second version.

9.3 BDPCA

Here BDPCA has been implemented using the new algorithm described in Section 6.3. The time lag d was calculated using the method from [Ku *et al.*, 1995] as proposed in the original article [Chen and Liu, 2002], even though it is probably not the best way to choose it, which is also discussed in Section 6.3. The calculation of the time lag is described in the section about DPCA in Chapter 6. Using this the time lag was selected to $d = 2$.

Based on the singular values the number of principal components was chosen to two out of the possible $(d+1)J = 12$. In Fig. 9.10 the monitoring of a batch with nominal parameters can be found. T^2 is inside its limits, while SPE has some spikes that fall outside. The spikes are probably because of interacting measurement noise between the $d + 1$ samples used at these times.

Fig. 9.11 shows the monitoring of a batch with 3% high C_{A0}. The first and second scores are outside their control limits at around 20 minutes and so is the T^2. They stay outside the limits for almost the rest of the batch duration. The SPE is flickering in and out of its control limit. Also here a dynamical change, now from the lower to upper control limit, takes places in the second score plot.

The monitoring of a batch with a ramp in E_1, up 1% from 90 to 145 minutes, is found in Fig. 9.12. The first score goes above its control limit at around 110 minutes and it stays outside the limit almost for the rest of batch. The SPE is mainly inside its control limits, but it is moving and out of its control limit for the last part of the batch duration.

When Figures 9.10, 9.11, and 9.12 are compared to the same figures for the second version of the variable-wise unfolding MPCA, i.e., Figures 9.7, 9.8, and 9.9, a lot of similarities can be found. If the sign of the second score plot in BDPCA is changed the score plots in the BDPCA method looks like a low-pass filtering of the one from the second version of the variable-wise unfolding MPCA. This is not surprising since the two methods are very similar and they are equivalent when $d = 0$. The filtering effect comes from that in BDPCA the latest $d + 1 = 3$ samples are used instead of only the latest sample.

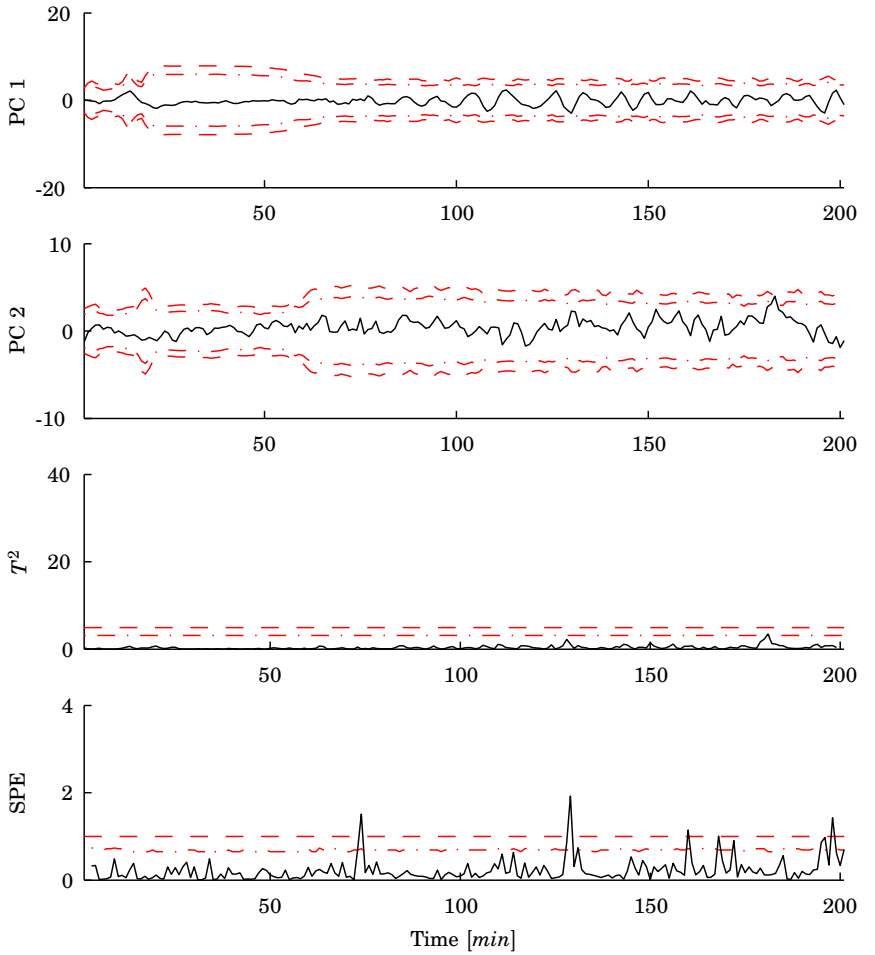

Figure 9.10 Monitoring of a batch with nominal parameters using BDPCA with $d = 2$.

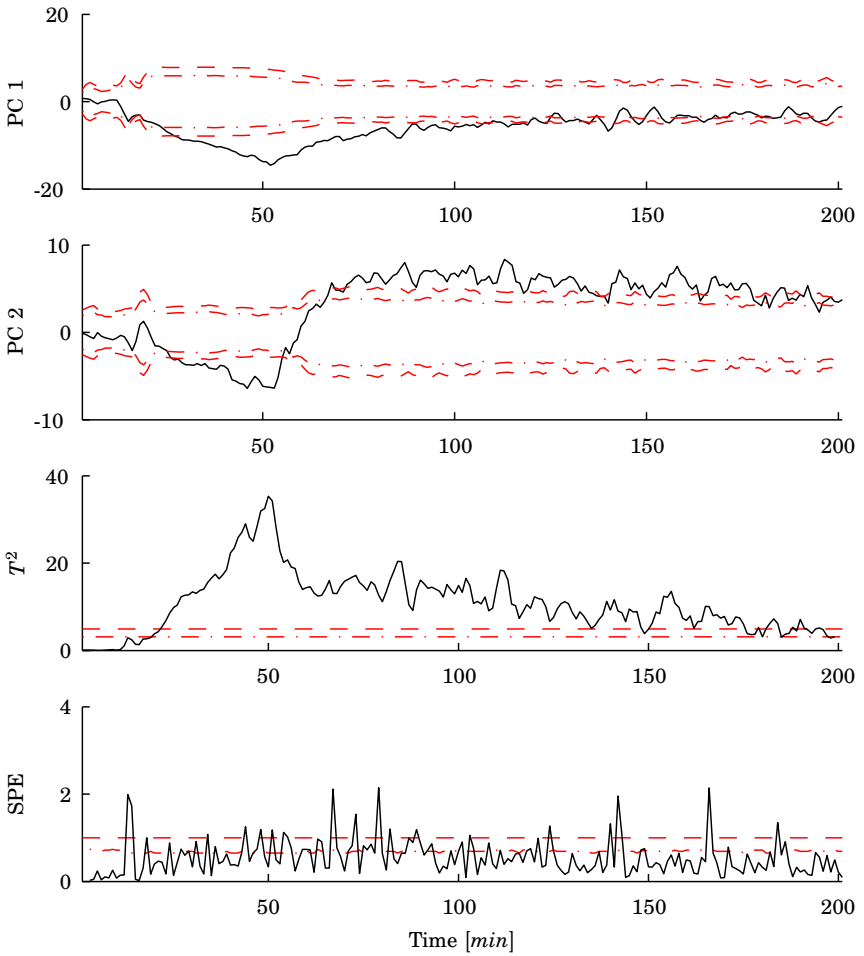

Figure 9.11 Monitoring of a batch with 3% high C_{A0} using BDPCA with $d = 2$.

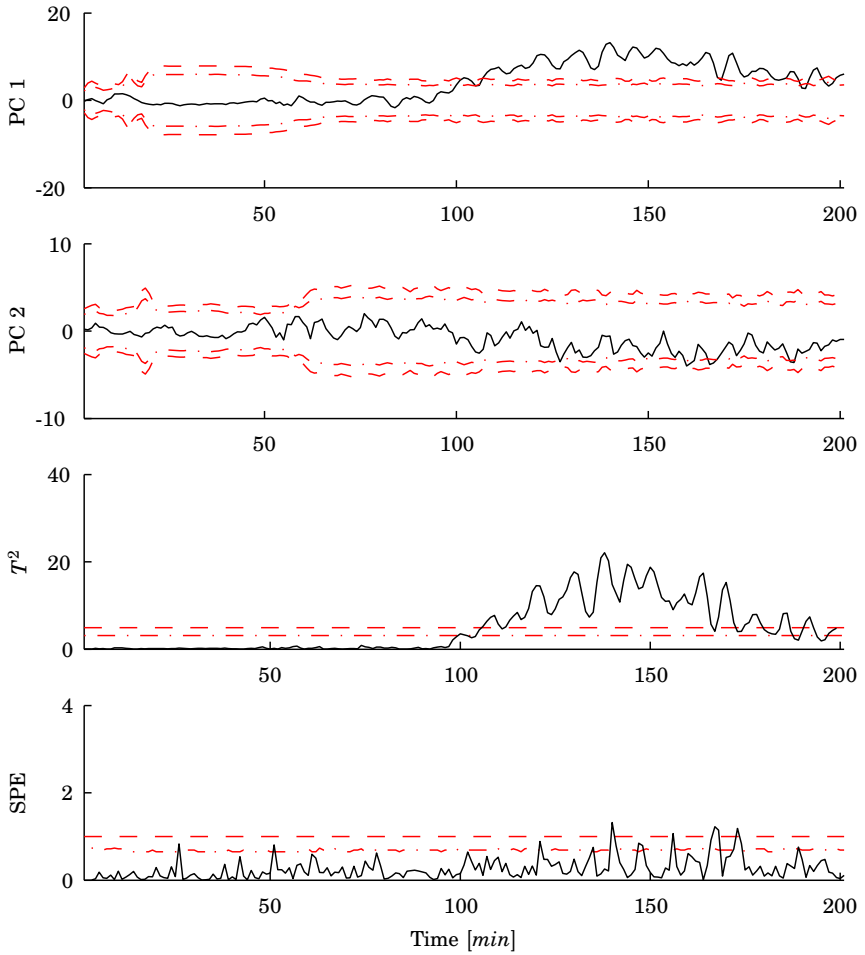

Figure 9.12 Monitoring of a batch with a 1% ramp in E_1 from 90 to 145 minutes using BDPCA with $d = 2$.

9.4 Multi Model MPCA

The results from an implementation of the multi model MPCA method where a model is calculated for each time instance is described here. The first step is to calculate how the singular values change over time, and thus with the models. This is done using SVD on the unfolded matrix $X^{I\times kJ}$ for each $k = 1, 2, \ldots, K$, i.e., at each sample time. The maximum number of singular values is either kJ or I depending on which has the smallest value.

In Fig. 9.13 the explained variance for the first 15 singular values at each sample time, i.e., for each model, are shown for the simulated batch process. After the switch between the heating and cooling phases, i.e., after around 20 minutes, the directions of the most significant principal components change since the two largest singular values change position at 25 minutes. This plot can be used to determine how many principal components that should be used in each model by looking at how many singular values are significant at each sample time.

A problem that may arise when calculating the singular vectors using SVD is that the sign of the singular vector may flip. This will give jumps in the score plots. This has been taken care of when calculating the models. An easy way to check the sign is to calculate the scalar product of the loadings for two models after each other in time. Some manual manipulation may also be necessary to achieve well suited monitoring plots for operators.

In Fig. 9.14, where a batch with nominal parameters is monitored, it can be seen how the number of principal components changes over time. From 5 to 15 minutes only one principal component is used and the second and third score plot show a value of zero. This is because according to Fig. 9.13 only one singular value is significant.

To make the plots of the scores more operator friendly the scores are kept in the same plot even though their significance changes. Thus, after 25 minutes the score of the first principal component does in fact correspond to the second most significant principal component since the singular values change place in Fig. 9.13. This has been used in all the figures concerning monitoring with this method. The T^2 99% limit is normalized to one since the control limit changes with the number of principal components used in the model.

The monitoring of a batch with deviation in C_{A0} can be found in Fig. 9.15. The score of the second principal component, or really the first, goes out of its control limits at around 35 minutes and so does the T^2. The SPE is mostly inside its control limit.

In Fig. 9.16 the monitoring of a batch where E_1 is ramped 1% from 90 to 145 minutes can be seen. The first plot to detect the fault is the

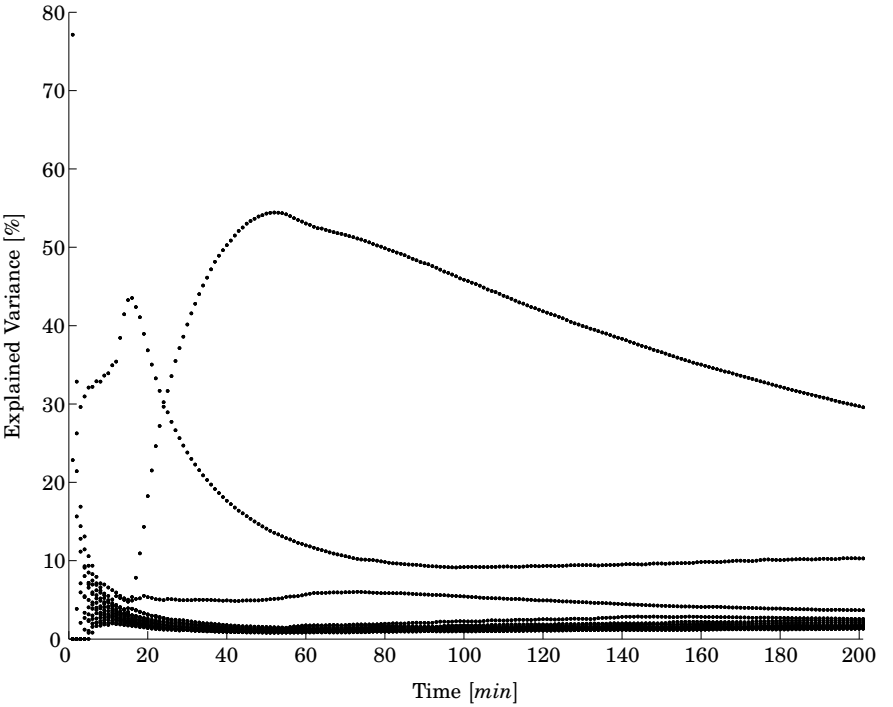

Figure 9.13 The explained variance calculated for the first 15 singular values of each unfolded matrix $X^{I \times kJ}$ in multi model MPCA.

SPE which goes out of its limit around 110 minutes. The score of the first principal component goes out its limit around 150 minutes and so does the T^2.

The multi model MPCA method gives results similar to the ones of batch-wise unfolding MPCA and at the end of the batch the two methods become equivalent and give the same model and the values in the plots are the same. An advantage of the multi model MPCA is that no filling in of future measurements is needed. On the other hand a substantial amount of work has been used in this implementation to get the right sign of the loadings and to reorganize the scores so that no jumps occur in the score plots when the significance of the principal components changes.

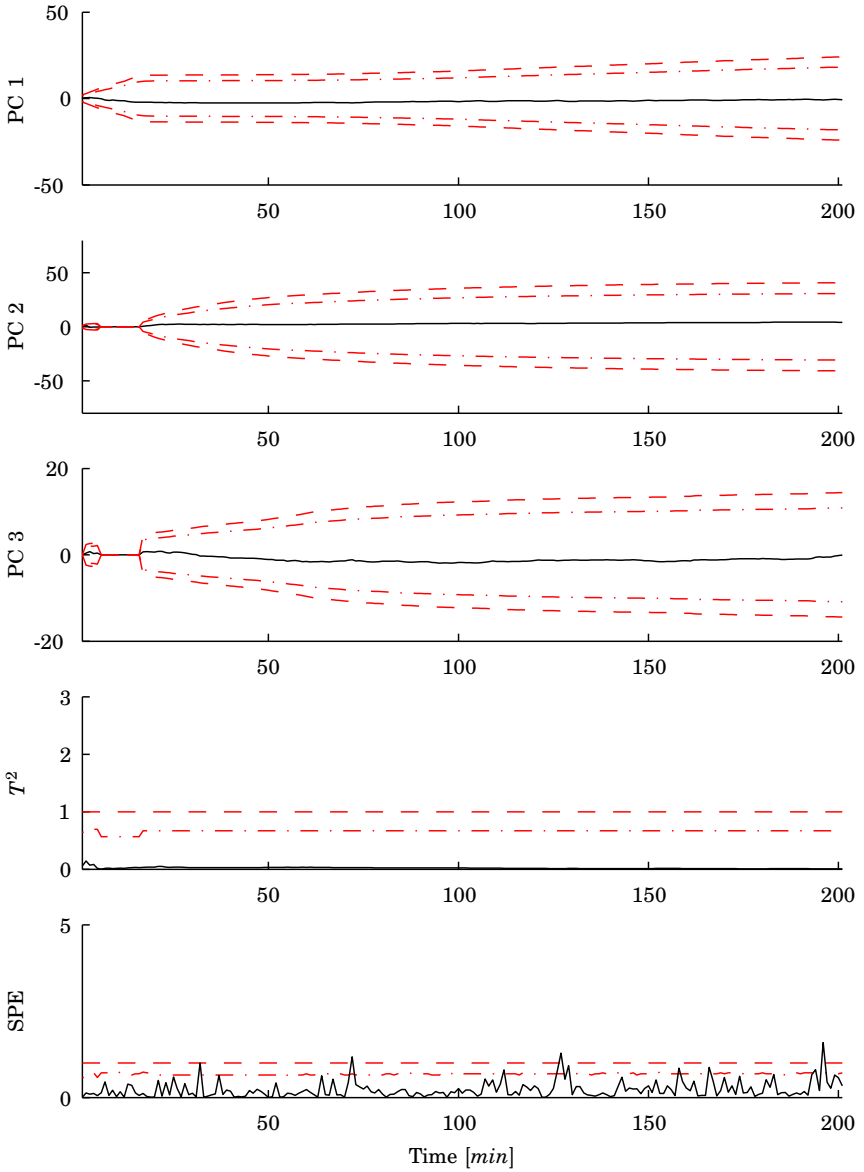

Figure 9.14 Monitoring of a batch with nominal parameters using multi model MPCA. T^2 has been normalized.

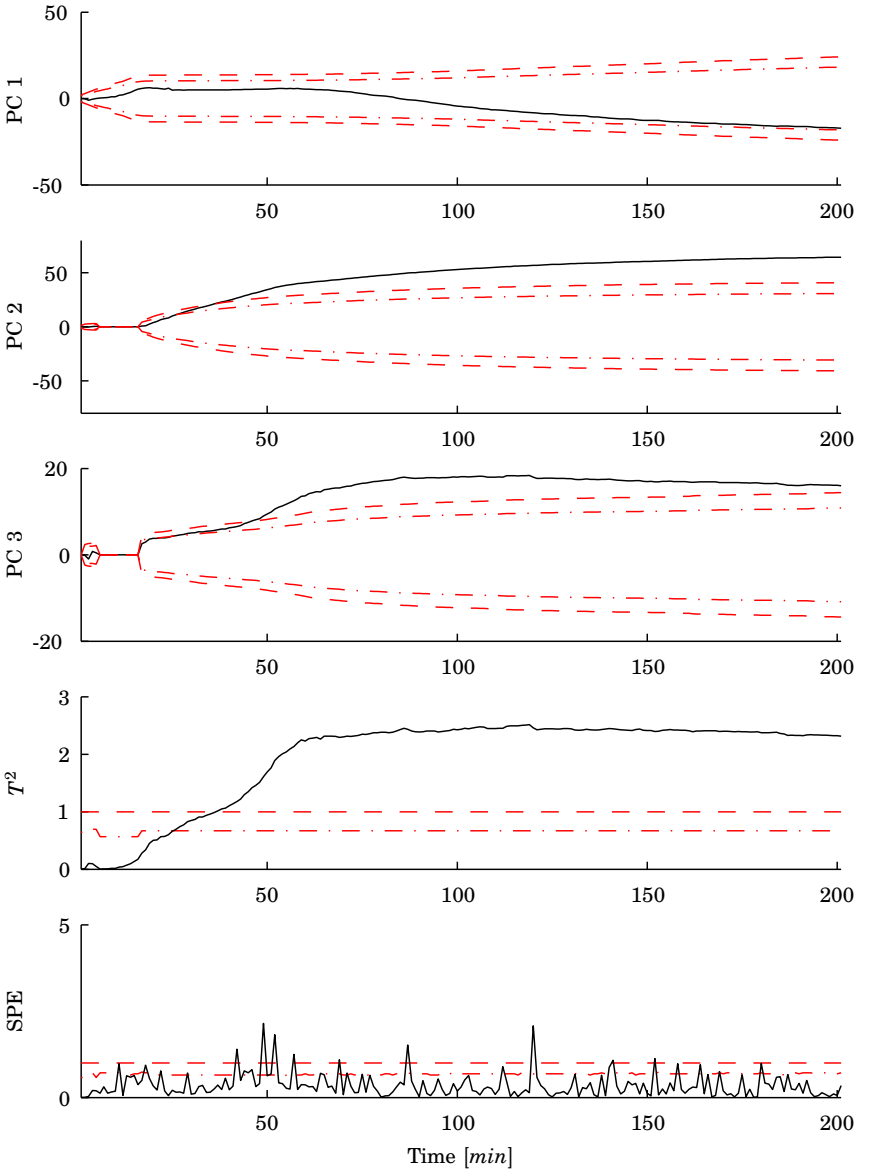

Figure 9.15 Monitoring of a batch with deviation in C_{A0} using multi model MPCA. T^2 has been normalized.

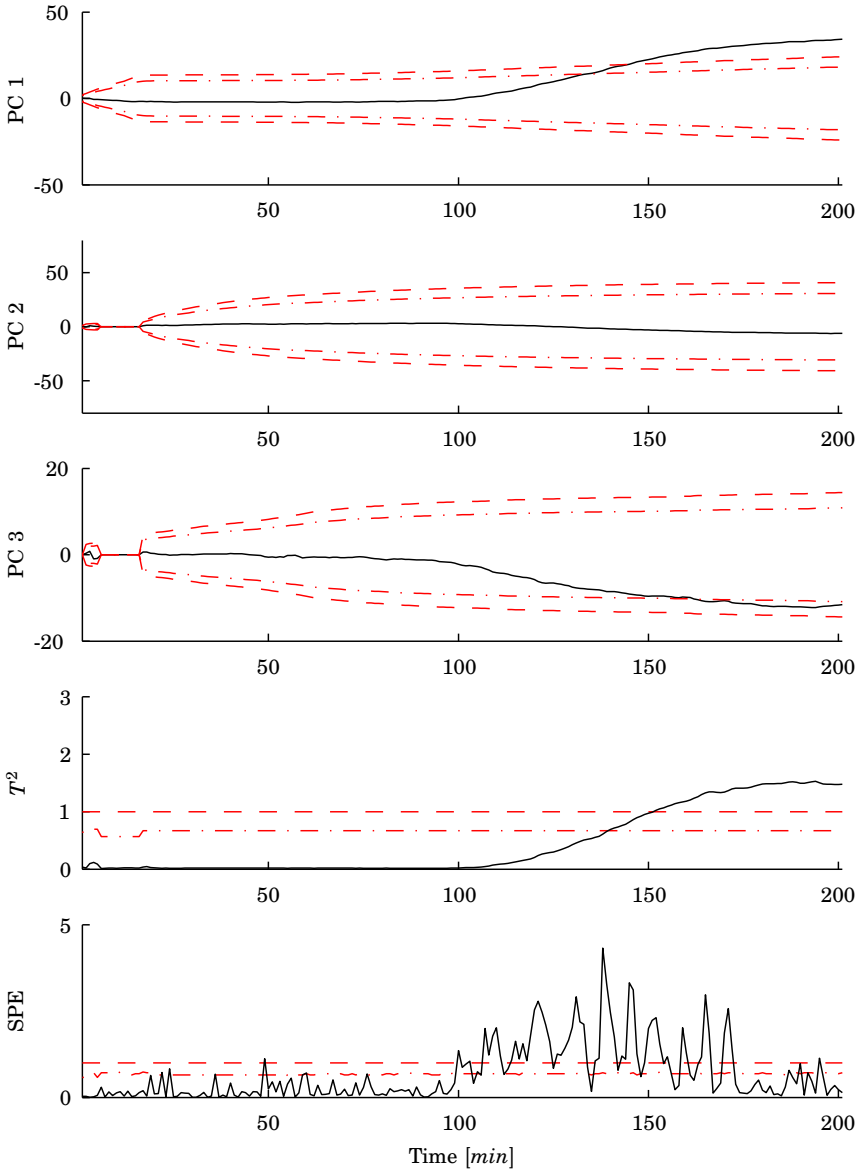

Figure 9.16 Monitoring of a batch with a 1% ramp in E_1 from 90 to 145 minutes using multi model MPCA. T^2 has been normalized.

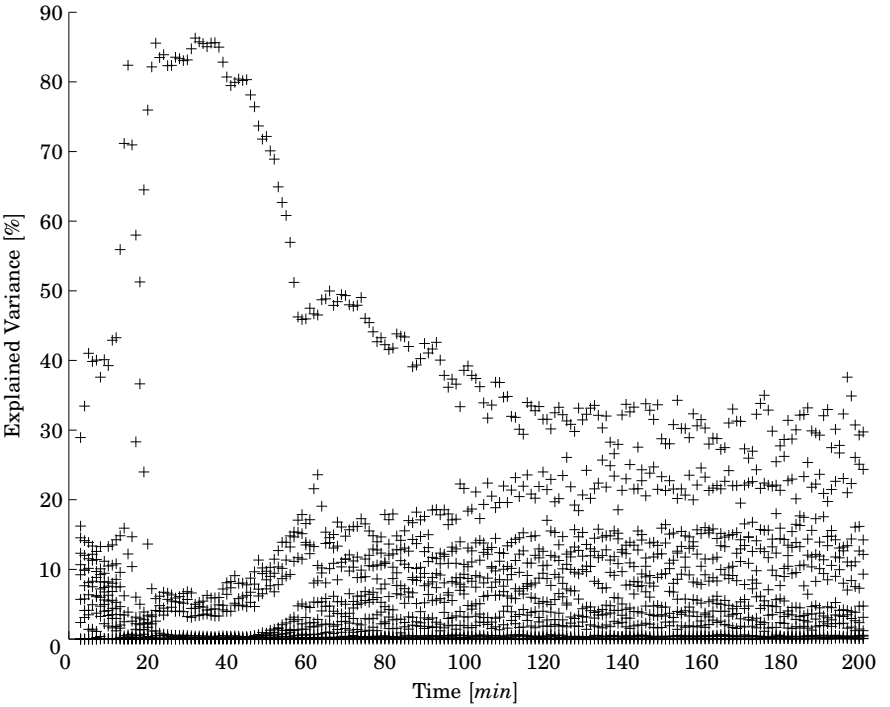

Figure 9.17 The explained variance calculated for the first 15 singular values of $X_d(k)$ at each time instance, with $d = 2$, in moving window PCA.

9.5 Moving Window PCA

The first step in moving windom PCA is to calculate the time lag d to be used for the models at the different time instances k. How this should be done is not described in any of the papers [Lennox *et al.*, 2001a; Lennox *et al.*, 2001b; Lennox *et al.*, 2002]. In these articles a window length $d = 5$ is used and it is claimed that varying d has limited impact on the results, e.g., page 274 in [Lennox *et al.*, 2002]. The reason for this can be explained by that when one uses batch-wise MPCA, i.e., here $X^{I\times KJ} = X^{70\times 804}$, the number of principal components is 3. This is for a system where the maximum number of principal components are either the number of batches, I, or the number of samples times the number of variables, KJ, depending on which is the largest, in our case I, which is 70 . The measurements are highly correlated in time and therefore the number of principal components will be low. Thus, different short windows will not change the performance drastically. The window length d has a filtering effect on the

models. If d is short the method is sensitive to noise and the loadings will be noisy. Here $d = 2$ is used, which maybe is a bit too sensitive to noise. Note that if the longest possible time window at sample k is used this will result in the multi model MPCA method.

The second step is to calculate the number of principal components to be used in the model. In Fig. 9.17 the explained variance for the first 15 singular values for $X_d(k)$ in Eq. 6.45 with $d = 2$ at each time instance k is shown. In Fig. 9.17 it is not as clear as in Fig. 9.13 that the singular values change significance at around 20 minutes since the plot is not as smooth.

To depict how the directions of the principal components, i.e., the values in the loadings, continuously change over time a set of batches without measurement noise was simulated with the same parameter variation as described in the beginning of the chapter.

The moving window PCA model at each sample time k can be written as

$$X_d(k) = T_R(k)P_R(k)^T + E_R(k) \tag{9.1}$$

where $T_R(k) \in \mathbb{R}^{I \times R}$ are the scores at time k and $P_R(k) \in \mathbb{R}^{(d+1)J \times R}$ are the loadings at time k, which consists of

$$P_R(k) = \begin{bmatrix} p_{1,1}(k) & \dots & p_{1,R}(k) \\ \vdots & \vdots & \vdots \\ p_{J,1}(k) & \dots & p_{J,R}(k) \\ \vdots & \vdots & \vdots \\ p_{dJ+1,1}(k) & \dots & p_{dJ+1,R}(k) \\ \vdots & \vdots & \vdots \\ p_{(d+1)J,1}(k) & \dots & p_{(d+1)J,R}(k) \end{bmatrix} \tag{9.2}$$

where R may change over time.

In Fig. 9.18 the values in the loadings of the first principal component associated with the first variable (i.e., $p_{1,1}(k)$, $p_{5,1}(k)$ and $p_{9,1}(k)$ since $d = 2$ and $J = 4$) are shown. It can be seen that there is a sharp change in the values at around twenty samples where the two singular values and thus the loadings change significance. A more smooth change is seen at around 60 samples. This can be related to the dynamic change of the second scores in Fig. 9.8 for the second version of variable-wise unfolding MPCA and Fig. 9.11 for the BDPCA methods. This also coincides with where the largest singular value in Fig. 9.13 is starting to increase after decreasing for some time indicating a change in the process. The reason why Fig. 9.18 is for the system without noise is that when noise is added

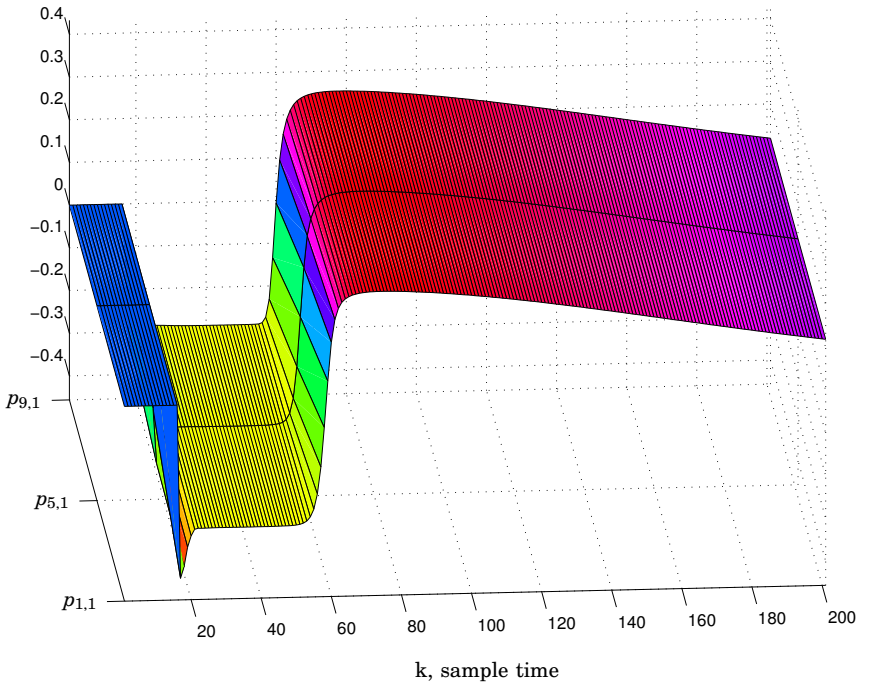

Figure 9.18 Values in the first loading vector for the first variable, $p_{1,1}(k)$, $p_{5,1}(k)$ and $p_{9,1}(k)$ for the noise free process.

the singular values and the loadings become "noisy", i.e., the principal components will jump from sample to sample and it becomes hard to see anything in the three dimensional plot.

Fig. 9.19 shows the monitoring of a batch with nominal parameters. Here can also be seen how the number of principal components changes over time. The batch stays well inside the limits of the score plots and T^2 plot. The SPE is also low except for one sample at the end.

Since only one component is used in the model between 22 and 90 minutes and the principal components change at around 20 minutes this will give rise to a transient in the score at this time. This transient is very clear in Fig. 9.20, where the deviation in C_{A0} first gives a high score in the first principal component and then it changes to a low value. Also the T^2 gives ambiguous information when it first almost goes outside its limit and then jumps back inside to once again go outside. This happens

because this is at the time of the switching between the heating and the cooling phase where the dynamics of the process changes.

In Fig. 9.21 the monitoring of a batch with 1% ramp E_1 from 90 to 145 minutes is shown. The score of the first principal component goes outside its limit at 110 minutes. The score of the second principal component is jumping back and forth outside the control limits and is not giving any good information.

The second score plot is in fact noisy in all the three figures, Fig. 9.19, 9.20, and 9.21. The reason for this is that the singular values in Fig. 9.17 are not distinct over time and therefore the loadings are not stable at all at the end and the directions jump between samples. The data consists mainly of noise at the end of the batch. Therefore, this method is not very well suited for this process.

9.6 Summary

In this chapter the results of implementing six different ways for multivariate statistical batch process monitoring have been presented. The methods are compared using two different faults considered to have impact on the final product. None of the methods have failed to detect any of the faults. Some of the methods have given very similar results, such as the variable-wise unfolded MPCA, second version, and BDPCA methods. Also the batch-wise unfolded MPCA and multi model MPCA methods show similar behavior in their plots for monitoring.

The time to detection of the faults in the different methods is now summarized. The first fault, a 3% high initial high concentration in C_A is detected first in the T^2 plots for all of the methods. The fastest methods for this fault are the ones using an unfolding of the data to preserve the variable mode, the two versions of variable-wise unfolded MPCA and BDPCA. Their time to detection is around 25 minutes. The moving window PCA method detects the fault at around 20 minutes when the T^2 detects the fault and then in the next sample jumps well inside the control limits to again go outside at 26 minutes. The methods preserving the batch mode, the batch-wise unfolded MPCA and the multi model MPCA, detects the fault at around 30 minutes. On the other hand the methods that preserve the variable mode tend to go back inside the control limits after the batch have been running for a while. It is important to understand that this does not mean that the batch can be considered as a good batch. It is enough that the batch have been outside the limits once in these methods. The methods preserving the batch mode stays outside the control limit since they take the whole batch duration into account when calculating the score and T^2.

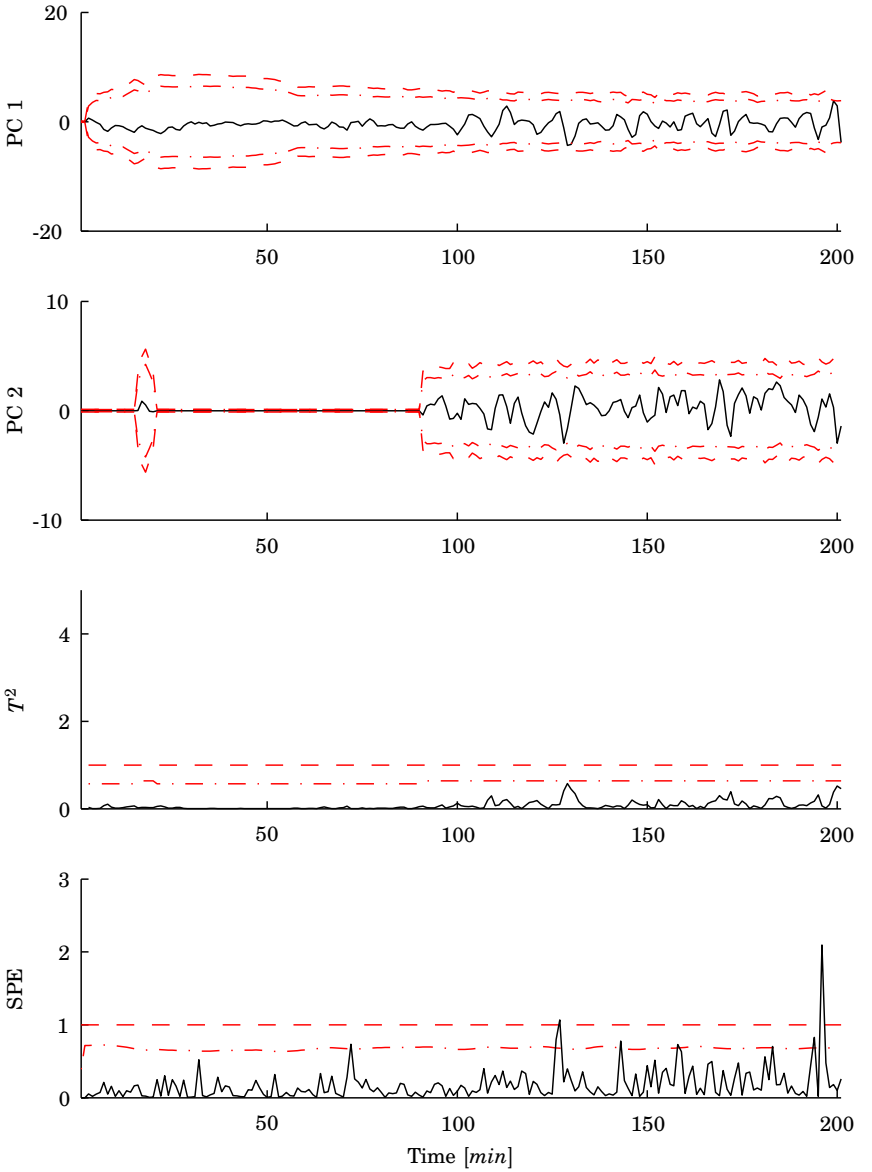

Figure 9.19 Monitoring of a batch with nominal parameters using sliding MPCA.

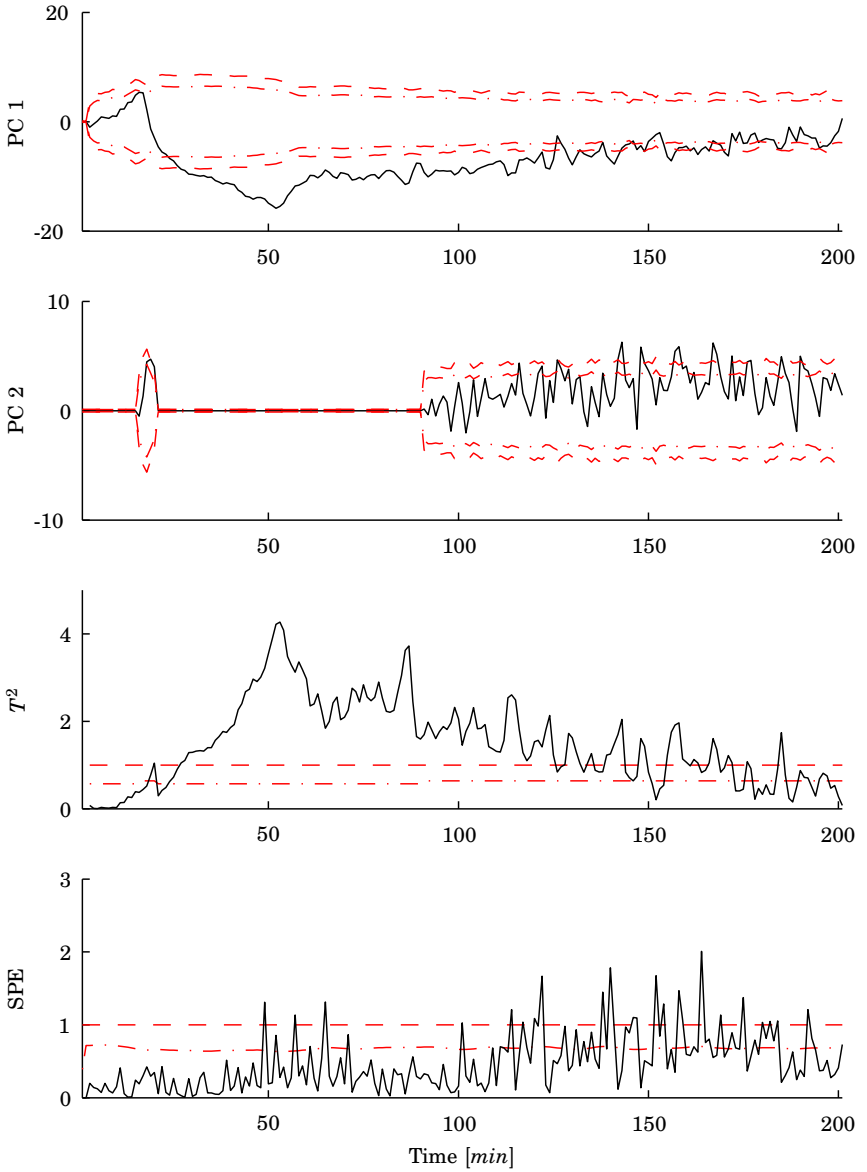

Figure 9.20 Monitoring of a batch with deviation in C_{A0} using sliding MPCA.

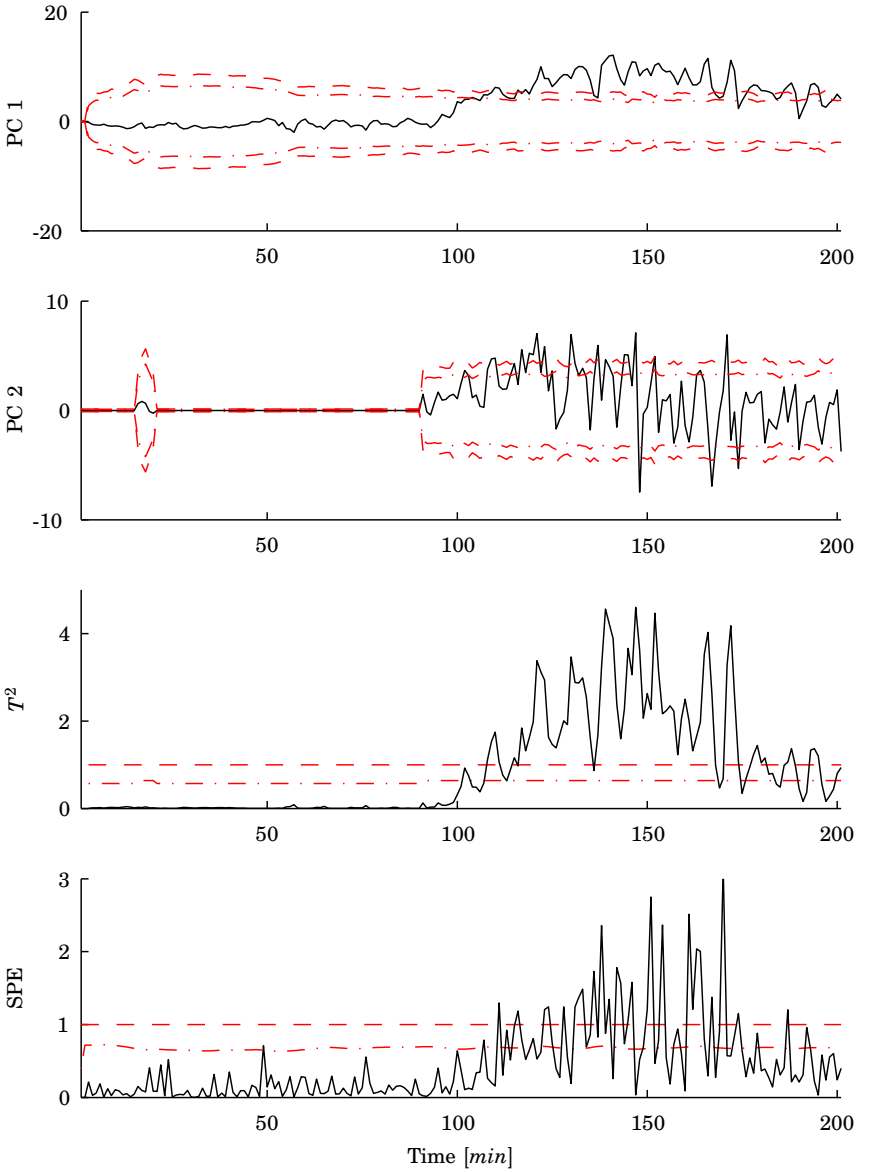

Figure 9.21 Monitoring of a batch with a 1% ramp in E_1 from 90 to 145 minutes using sliding MPCA.

The second fault, a 1% ramp in the activation energy E_1 from 90 to 145 minutes, is first detected in the SPE for the methods preserving the batch mode and in the T^2 in the methods preserving the variable mode. The time of detection is at around 105 minutes for the batch-wise unfolded MPCA, second version of variable-wise unfolded MPCA, multi model MPCA, BD-PCA, and moving window PCA. The only method that is a bit slower is the first version of variable-wise unfolded MPCA, which has the worst performance moving in and out of the limits and detecting the fault after as long as 116 minutes. In all the methods the plot that first detects the fault is moving in and out of the control limits. Again the methods preserving the variable mode have a tendency to move back into the normal region at the end of the batch. It can be noticed that SPE for the batch-wise unfolded MPCA in Fig. 9.3 has very similar characteristics to T^2 of second version of variable-wise unfolded MPCA in Fig. 9.9 from 90 to 140 minutes (except for the different scale on the y-axis). This can be explained by that the data is mean centered and scaled in the same way and both the indices are based only on the latest sample.

The moving window PCA method seems not to be suited for this process due to that the singular values changes significance in an unfavorable way over the duration of the batch. The method is also not suited because at the end of the batch very little structured information is available, the data is almost only noise. This can be seen by looking at the singular values at the end of the batch duration in Fig. 9.17, where they are getting closer and closer to each other and it is impossible to see if they change significance.

The Methods Revisited

As stated above the differences between the first and the second version of the variable-wise unfolded MPCA, is the centering and scaling of $X^{IK\times J}$. In the first version the grand mean and standard deviation over all times and batches is used. In the second version the mean and standard deviation at each sample time is used instead. To understand what this means a small example with dummy data is used.

In Fig. 9.22 data is shown that could originate from a batch process with two measured variables, x_1 and x_2, i.e., $J = 2$. The number of batches is $I = 15$ and the number of samples is $K = 20$. Each small ellipse contains the measurements from the I batches at each sample time, showing how the relationship between the two variables are changing over time, i.e., the direction of the semi-axes of the ellipses changes. Data from the simulated batch process in Chapter 7 has similar characteristics to this example.

The grand mean of the data, used in the first version of variable-wise unfolded MPCA, is marked in the figure with a $*$. The means used in the

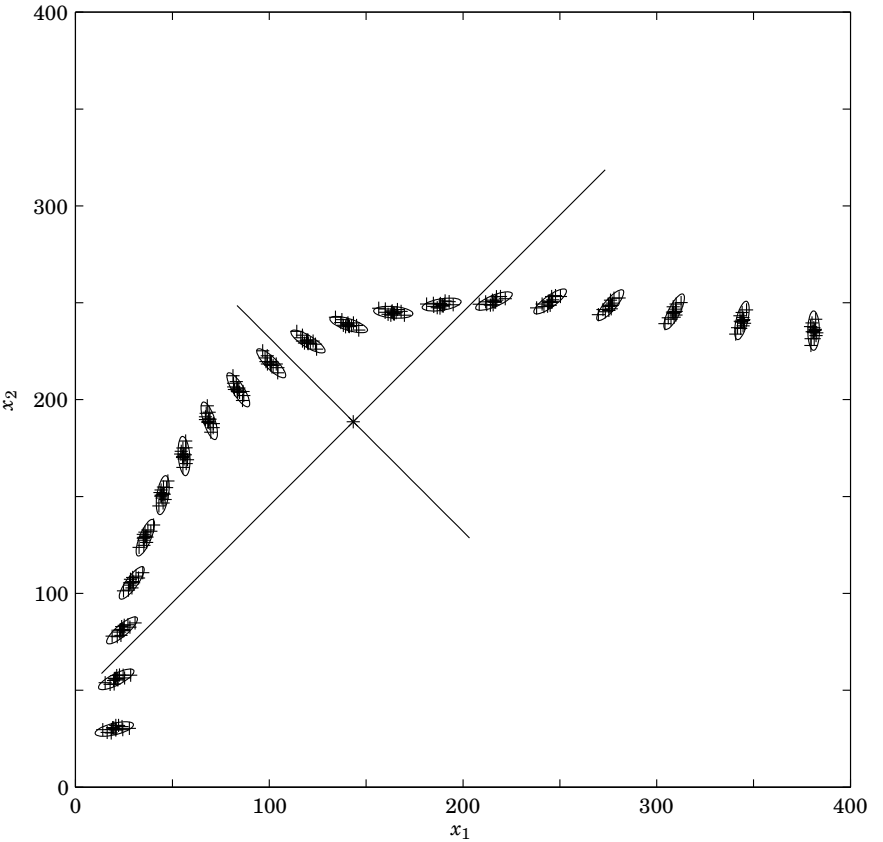

Figure 9.22 Dummy data used for discussion of the methods. Each value is marked by a +. The grand mean is at the intersection of the two lines.

second version of variable-wise unfolded MPCA are the center points of each ellipse. After scaling the data to unit variance the principal components will point in a 45^o angle to the original variables, shown by the lines in the figure. This means that there will be points in time where the monitoring will work better than at others, depending on if the semi-axes of the small ellipse are aligned with the principal components or not. If they are aligned the confidence limits will be tight and if they are not aligned the limits will not be tight. The reason for this is the recalculation of the limits using the matrix of the scores in Eq. 6.26.

A variant of the problem with more or less tight limits arises in the second version as well. This is because the rotation of the small ellipses

in Fig. 9.22 will make the limits more or less tight over time. After the data is centered and scaled within each ellipse the data will still have different rotation when used in the matrix $X^{IK\times J}$. The moving window PCA method makes the limits tighter since the models are calculated at each sample time, i.e., only the data inside each small ellipse is used. The ellipses in Fig. 9.22 are in fact the 99% limits at each sample if the time lag $d = 0$. On the other hand the problems discussed above for the moving window PCA method still exist.

In the batch-wise unfolded MPCA methods the measurements over time are seen as different correlated variables and the model has a weight for each variable at each time in the loadings.

Normal Variation during Faults

In this chapter only one fault is present in the faulty batch at the time, while all other parameters at kept at their nominal values. This seems to be praxis in most articles in the area of statistical monitoring. Since one of the assumptions is that there are batch to batch variations in several parameters a thorough evaluation of a method should be done using several faulty batches with the same fault but the other parameters varying with its normal deviations from the nominal values using Monte-Carlo simulation. This is since different parameters can push the process in the opposite direction and even cancel out the effect from the fault. During the evaluation of the methods in this thesis it has been noticed that the time to detection of a fault may increase (or decrease) significantly when using parameters that are large but within the normal variation instead of having their nominal values. Also multiple faults occurring at the same time can have similar effects.

10

Combining Model-Based Estimation and Multivariate Statistical Methods for Batch Process Monitoring

10.1 Introduction

As described in earlier chapters there exist a huge number of different methodologies for fault diagnosis. A description of some of the fundamental and practical differences between statistical and causal model-based approaches to fault detection and isolation can be found in [Yoon and MacGregor, 2000]. One of the major differences is that a first-principles model describes the causal effects from inputs to outputs, while a statistical model only describes the covariance structure of the measured variables during normal operation. In this chapter a new way of combining the use of a first-principles model of a batch process to estimate unmeasured states and parameter values together with multivariate statistical methods using historical data from previously successful batches for on-line fault detection is studied. A simplified version of the model described in Chapter 7 is used for the discussions and the evaluation of the new method, see Section 10.3. Normal batch to batch variations are assumed in the heat transfer coefficient h_i, activation energy E, and initial concentration of A, i.e., C_{A0}. Different faults associated with the same parameters are considered; decrease in the heat transfer coefficient inside the reactor (i.e. fouling), changes in the activation energy (a ramp or a large initial deviation), and abnormally large deviations in C_{A0}.

Several of the methods described in Chapter 5 have been combined to hopefully increase the robustness and efficiency of the diagnosis methods. For example, in [Gertler and McAvoy, 1997; Gertler *et al.*, 1999] PCA and parity relations are combined to increase the isolation properties of the faults and in [Ku *et al.*, 1994] parallel Kalman filters and Hotelling's T^2 statistics are combined. In [Vedam and Venkatasubramanian, 1999] PCA is combined with SDG. PCA is used for detection of abnormal process behavior and the contributions from the measured variables are fed to the SDG to perform diagnosis to find the root cause of the fault. In [Ramaker *et al.*, 2002] a discussion takes place on how to incorporate external information into multivariate statistical methods. The initial conditions of the batch run, Z, can be added to the measurements. A grey model approach [Gurden *et al.*, 2001] for batch process monitoring is mentioned but not fully described. Grey models are hybrid models combining known (white part) and unknown (black part) causes. The model is estimated as a single least squares optimization where the parameters are calculated simultaneously. In [Yoon and MacGregor, 2001b] different ways of incorporation of external information is also described.

In [Bonné and Jørgensen, 2004; Gregersen, 2004] a method for dynamic input-output modeling of batch processes is described. The model is developed by identifying a grid of interdependent linear time invariant autoregressive models with exogenous inputs, i.e., ARX models, using Tikhonov regularization/ridge regression [Tikhonov, 1963; Hoerl and Kennard, 1970]. The model is based on using the trajectory from initial condition to sample $K-1$ from the ith batch $y^i \in \mathbb{R}^{KN}$

$$y^i = \left[y^i(0)^T \dots y^i(k)^T \dots y^i(K-1)^T\right]^T, \tag{10.1}$$

where $y^i(k)^T = \left[y_1^i(k)^T \dots y_N^i(k)^T\right]^T$ are the measurements of the N outputs at sample time k from the ith batch. The measurements from all I batches are then used to form the model

$$qX = AX + BU + e, \tag{10.2}$$

where q is the forward shift operator in time, $X = \left[y^1 \dots y^I\right] \in \mathbb{R}^{KN \times I}$ is formed of the measurement trajectories from the I batches, $U = \left[u^1 \dots u^I\right] \in \mathbb{R}^{KM \times I}$ is formed in a similar way from the trajectories of the M inputs from the I batches, $A \in \mathbb{R}^{KN \times KN}$ and $B \in \mathbb{R}^{KN \times KM}$ are structured lower block-triangular matrices, and e is the residual. Eq. 10.2 is solved by least squares estimation, while at the same time the structure of A and B is enforced by Tikhonov regularization, which ensures causality and that the grid of models becomes interdependent. In [Gregersen, 2004] the model matrices A and B are suggested to be used for filtering in combination

with batch-wise unfolded MPCA, but it is not implemented or discussed in depth. Instead the model is used in univariate SPC using CUSUM plots for a one-step ahead predictor. The data used in the method is the same as in MPCA but the variables need to be divided into inputs and outputs prior to the parameter estimation.

Using a similar data structure as in Eq. 10.1 the article [Lee and Dorsey, 2004] uses the subspace identification method N4SID [Van Overschee and De Moor, 1994] to identify a dynamic batch-to-batch model. The model is used together with PCA to monitor the process for batch-to-batch variations using T^2 and SPE. An on-line approach is also described where a periodically time-varying (PTV) Kalman filter is used.

In this chapter drawbacks of some of the existing methods are highlighted and discussed. In [Ramaker *et al.*, 2002] it is concluded that the potentials of combining methods to incorporate external information is not yet fully explored and that further research is needed in this area.

10.2 Model-Based PCA

A previous method where a first-principles model and a multivariate statistical method are combined is model-based principal component analysis, MBPCA, which is described in [Wachs and Lewin, 1998; Wachs and Lewin, 1999; Rotem *et al.*, 2000; Rotem and Lewin, 2000]. In MBPCA data simulated from a first-principles model with nominal parameters are subtracted from the measurements of the real process and then PCA is applied to the difference between the two. This can be seen as the calculation of a set of residuals. The aim of MBPCA is to eliminate features in the output from the process that can be explained by a model of the process. Thus, any features unexplained by the model, e.g., effects of disturbances, and changes in system behavior due to faults, will appear in the residuals.

The procedure is described in both open loop, see Fig. 10.1, and in closed loop, see Fig. 10.2. In the open-loop version the control signals to the real process are also fed to the model to simulate normal behavior of the process and the approach is only applicable if the process is stable. Despite this it is used for monitoring of the exothermic, and thus unstable, batch reactor in an example in [Wachs and Lewin, 1998]. The example is the batch rector model with the simplified heating, i.e., no steam, described in Chapter 7. This simplification is further described in the next section. The method used for the monitoring is MPCA using variable-wise unfolding [Wold *et al.*, 1998] described in Chapter 6.

In the closed-loop version the model is controlled by a controller with the same parameters as the controller in the real process and following

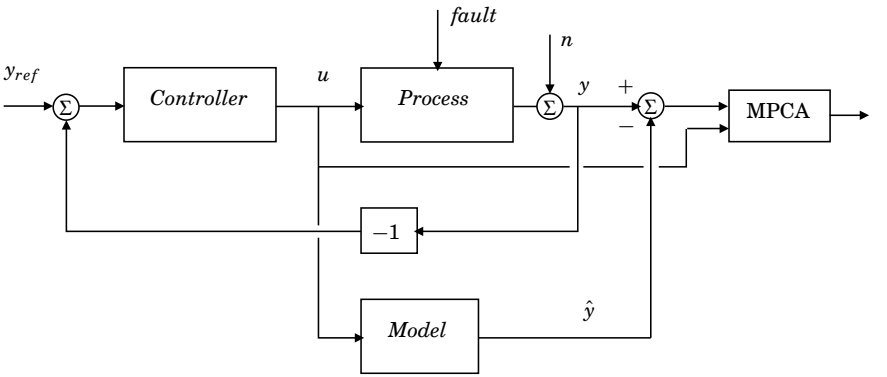

Figure 10.1 MBPCA in the open-loop approach.

the same reference trajectory. This method is used in [Rotem *et al.*, 2000] for monitoring of a compressor. The compressor is a continuous system, which is affected by a periodical disturbance making the process change its operating point. By modeling the change of the operating point, the fault detection abilities are improved. This can be seen as if the reference trajectory of the closed loop system is changed. This is different from the case of monitoring a batch process where the reference signal is the same over time from batch to batch. Therefore, the simulations from the model gives the same predictions of the process for every batch and it will not add any extra information to the diagnosis. The subtraction of the simulations from the model in closed-loop MBPCA is equal to mean centering the data in the batch-wise MPCA online. Thus the closed-loop version of MBPCA is equal to the second version of variable-wise MPCA, which is described in Chapter 6.

An extension of MBPCA can be found in [McPherson *et al.*, 2001; McPherson *et al.*, 2002] where an extra error model is added to remove structures in the residuals. How this is done is only vaguely described. It seems like the open-loop approach is used in the work, which is not suitable for an unstable process.

Simplifications of the Reactor Model in MBPCA

It should be noted that in the articles [Dong and McAvoy, 1995; Wachs and Lewin, 1998; McPherson *et al.*, 2002; Chen and Liu, 2002] the model of the batch process described in Chapter 7 has been simplified. The simplification of the model concerns the jacket during the heating phase. Instead of using a two-phase model for the steam, which is very hard to model, it

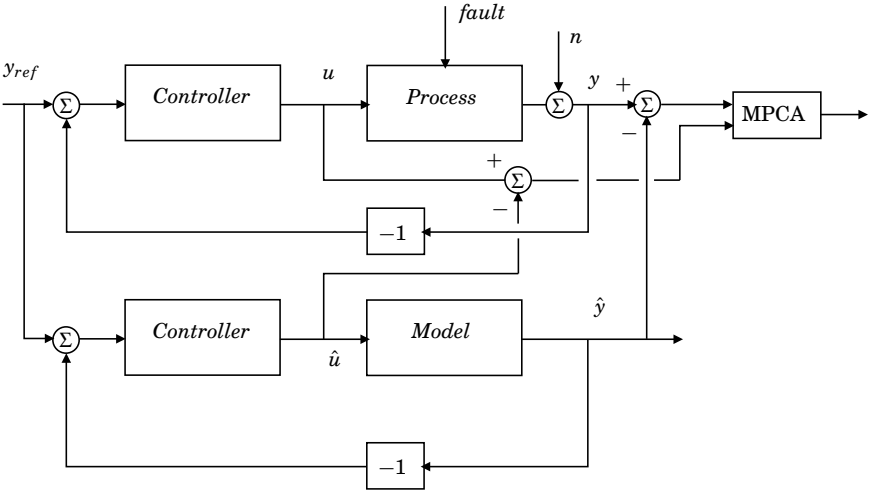

Figure 10.2 MBPCA in the closed-loop approach.

is assumed that the heating and cooling media is a fluid with the same density ρ, heat capacity C_p, heat transfer coefficient h_o, and that these are all independent of temperature. Another assumption is that the jacket is always full with fluid.

In none of the papers describing MBPCA the issue that the concentrations in the reactor are not measurable, and therefore need to be estimated, is discussed. The concentrations are simulated in open loop assuming the initial concentration of A is known and not subject to normal batch to batch variation which is assumed here. As will be described in the next section batch to batch variations in C_{A0} makes the problem much harder.

10.3 Fundamental Properties of a Batch Process

To be able to better understand the fundamental properties of a batch process a simplified version of the model described in Chapter 7 will be used here. The reaction is simplified to one reaction and it is assumed that the amount of energy transferred to and from the reactor, Q_M, can be measured. The process will now consist of only one irreversible exothermic reaction according to

$$A \xrightarrow{k} B \tag{10.3}$$

The mass and heat balances for this process are given by the following equations

$$\begin{aligned} \frac{dC_A}{dt} &= -kC_A \\ \frac{dT}{dt} &= \frac{-\lambda}{\rho C_P} kC_A + \frac{Q_M}{V\rho C_P} \end{aligned} \tag{10.4}$$

where again C_A is the concentration of A in the reactor, k is the reaction rate given by the Arrhenius expression

$$k = \alpha \exp\left(\frac{-E}{R(T+460)}\right) \tag{10.5}$$

where α is a constant, and E is the activation energy. R is the universal gas constant and T is the absolute temperature in the reactor. λ is the heat of reaction, ρ is the density of the contents of the reactor, and C_P is the heat capacity of the contents.

A reference temperature profile for a batch was defined with a starting temperature of 80^oF, continuing with a ramp up to 160^oF from 25 to 125 samples with a sampling time of one minute. The temperature is kept constant until 200 samples when the temperature is slowly ramped down. The profile has been chosen to have a similar shape as the one described in Chapter 7.

Recursive Least Squares Estimation

To try to separate the different faults from each other it can be seen that with the simplifications of the jacket used in MBPCA, described above in Section 10.2, the heat transfer coefficient inside the reactor, h_i, can be estimated using a recursive least squares, RLS, estimate [Åström and Wittenmark, 1995]. By using the heat balance for the reactor from Eq. 7.2 and 7.5

$$\begin{aligned} \frac{dT_M}{dt} &= \frac{Q_M - Q_J}{\rho_M C_M V_M} \\ Q_M &= h_i A_i (T - T_M) \\ Q_J &= h_o A_o (T_M - T_J) \end{aligned} \tag{10.6}$$

and rewriting them to fit the structure given by

$$y(k) = \varphi^T(k)\theta \tag{10.7}$$

where y, φ, and θ are given by

$$\begin{aligned} y(k) &= \frac{dT_M(k)}{dt}\rho_M C_M V_M + h_o A_o (T(k) - T_M(k)) \\ \varphi^T(k) &= A_i (T(k) - T_M(k)) \\ \theta &= h_i \end{aligned} \tag{10.8}$$

The recursive least squares estimate $\hat{\theta}$ with forgetting factor λ is now given by

$$\begin{aligned} \hat{\theta}(k) &= \hat{\theta}(k-1) + K(k)\left(y(k) - \varphi^T(k)\hat{\theta}(k-1)\right) \\ K(k) &= P(k-1)\varphi(k)\left(\lambda I + \varphi^T(k)P(k-1)\varphi(k)\right)^{-1} \\ P(k) &= \left(I - K(k)\varphi^T(k)\right)P(k-1)/\lambda \end{aligned} \tag{10.9}$$

Using the estimate of h_i, Q_M can be calculated and the fault detection can be reduced to detecting the faults in the activation energy and the initial concentration of A.

Extended Kalman Filtering

Again, only the temperature in the reactor is assumed to be measurable, while the concentration needs to be analyzed in a laboratory and is not available until after the batch is finished. The concentration is assumed to be controlled in an open-loop fashion by letting the temperature in the reactor follow an optimal pre-specified trajectory for the nominal process. If the concentration was measurable the problem would be very much different. Then of course the concentration would be controlled with the temperature in a cascaded fashion. The usual way to try to circumvent this problem in automatic control is to use an observer to estimate the unmeasured state and then use this for feedback control. Since the batch process is a non-linear system the unmeasured concentration can be estimated using an extended Kalman filter EKF, see e.g., [Russell *et al.*, 2000] where an EKF is used for monitoring of batch processes. EKF is a way to apply the linear Kalman filter theory to a non-linear system by linearizing the model around a nominal trajectory. The algorithm is as follows. A nonlinear process is given by

$$\begin{aligned} \frac{dx(t)}{dt} &= f(x(t), u(t)) \\ y(k) &= g(x(k)) \end{aligned} \tag{10.10}$$

where x is the state variable, y is the output, and u is the input. Continuous time is denoted with t and discrete time with k. Assuming that the input u is constant over every sampling period the solution becomes

$$x(k) = x(k-1) + \int_{(k-1)t_s}^{kt_s} f(x(t), u(k-1))dt = F(x(k-1), u(k-1)) \tag{10.11}$$

where t_s is the sampling time. To also describe effects of state and measurement noise the discrete time model now is given by

$$\begin{aligned} x(k) &= F(x(k-1), u(k-1)) + \omega(k-1) \\ y(k) &= g(x(k)) + \nu(k) \end{aligned} \tag{10.12}$$

where $\{\omega(i)\}$ and $\{\nu(i)\}$ are assumed to be zero-mean uncorrelated sequences of random vectors with covariance Q and R respectively. The EKF then becomes

$$\begin{aligned}
\hat{x}(k+1|k) &= F(\hat{x}(k|k), u(k)) \\
P(k+1|k) &= A(k)P(k|k)A^T(k) + Q \\
K(k+1) &= P(k+1|k)C^T(k+1)\big(C(k+1)P(k+1|k)C^T(k+1) + R\big)^{-1} \\
\hat{x}(k+1|k+1) &= \hat{x}(k+1|k) + K(k+1)\left(y(k+1) - g(\hat{x}(k+1|k))\right) \\
P(k+1|k+1) &= (I - K(k+1)C(k+1))P(k+1|k))
\end{aligned} \tag{10.13}$$

with

$$\begin{aligned}
C(k) &= \left.\frac{\partial g(x)}{\partial x}\right|_{x=\hat{x}(k|k-1)} \\
A(k) &= \left.\frac{\partial F(x,u)}{\partial x}\right|_{x=\hat{x}(k|k-1),u=u(k)}
\end{aligned} \tag{10.14}$$

where P is the covariance associated with the estimate of the state $\hat{x}$. The filter needs the starting conditions $P(0|0)$ and $\hat{x}(0|0)$.

The estimated concentration will converge to the real concentration depending on the initial state and covariance matrix in the EKF. In a batch process the initial state may be very important. The initial concentration can be estimated recursively by using a fixed-point smoother using the solution from the EKF [Russell *et al.*, 2000]. Another way of estimating the state of a non-linear system is the unscented Kalman filter UKF [Julier *et al.*, 1995] where the approximation by linearization is avoided.

The EKF will only give the true estimate of the concentration if the parameters are constant and this is not true in this framework. The parameters are assumed to instead have a normal variation. If the activation energy E is varying randomly between batches due to different amount of impurities present in different batches the concentration can no longer be accurately estimated due to the mismatch between the model and the process.

If E is assumed to be constant in each batch one way to estimate both the activation energy and the concentration is to augment the EKF with a new state. This is called the augmented EKF, AEKF. The problem is that the system is not observable and all the states cannot be estimated. This can be seen by linearizing the system around the nominal trajectories and applying linear methods for observability [Khalil, 2002] or by looking at local decompositions of the system [Isidori, 1995]. Thus this is not a suitable approach.

As have been described even this simple exothermic batch process is very complex. The fundamental properties of this process is that it is non-

linear and time varying. It is unstable since the reaction is exothermic and need to be controlled to not to cause a runaway reaction. It is unreachable, since the reaction is irreversible, and it is unobservable when normal variation is assumed in parameters.

10.4 Combining Model-Based Estimation and Multivariate Statistical Methods

The question tried to be answered here is how knowledge about the process in the form of a first-principles model (with uncertainties in the model parameters) can be combined with historical knowledge? How can the measured variables, inputs and outputs, from a historical data base in cooperation with the model improve the fault diagnosis when the process has the properties described above? The approach taken here is to try to find a function $f(y,u)$ that is able to describe some underlying properties of the system, which can improve the detectability of faults in the process when added to the measured variables and fed to any of the multivariate statistical methods described in Chapter 6. A schematic of this approach can be found in Fig. 10.3.

In several publications it has been proposed that material and energy balances can be used as extra variables, e.g., in [Yoon and MacGregor, 2001b; Kourti, 2003b], and that the measurement matrix, $\underline{X}$, should be augmented with the extra variables. This approach was tried on the simulated batch process in Chapter 7, but no improvements were seen. Then

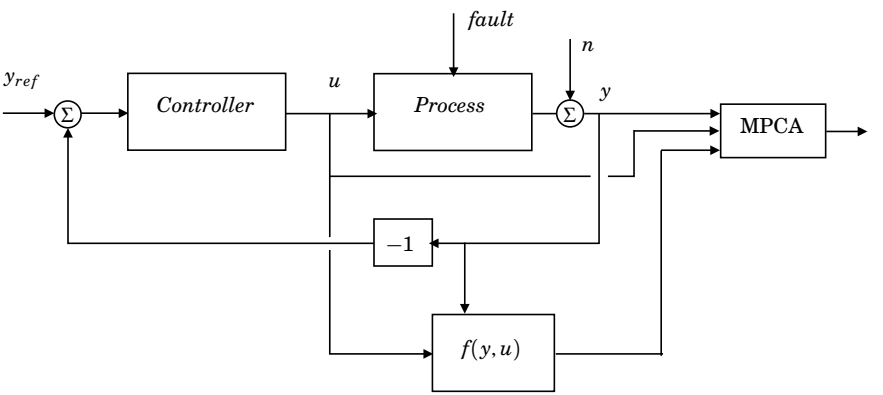

Figure 10.3 General description of the combination of combining model-based an multivariate statistical methods for batch process monitoring.

the systematic approach for generating residuals in the DMP method, see Chapter 8, was used to augment the measurement matrix with these residuals. This did surprisingly not improve the fault detection either. One reason might be that the material and energy balances, and the residuals from DMP are mainly linear combinations of the original variables and that adding these to the measurement matrix will only act as weights of the original variables when calculating the model using PCA.

The new idea developed here is that even though an estimation of either C_A or E, in the example used here, will not give correct estimates it will in some sense still describe the relations between the states in the system. It will describe a "subspace" depending on the normal parameter variations. This in combination with a historical data base where the same estimations have been performed will increase the ability to detect an abnormal state in the process. Below two approaches for this using an RLS for estimating E and an EKF for estimating C_A are presented. The two approaches are tested on the simplified batch process simulation. It is assumed that the heat transfer coefficient inside the reactor, h_i, is known.

RLS Approach

The first approach is to estimate the activation energy, E, using an RLS estimator. It will be shown that this estimation gives useful results in it self even though the true parameter cannot be estimated when C_{A0} does not have its nominal value.

The temperature in the reactor, T, is measured and Q_M can be calculated from the measurement of T_M using Eq. 10.6, when the heat transfer coefficient inside the reactor, h_i, is considered to be known. With these two an estimate of the change in C_A, $\frac{dC_A(k)}{dt}$, can be calculated by rewriting the equations in Eq. 10.4 to

$$\begin{aligned} \frac{d\overline{C}_A(k)}{dt} &= -k\overline{C}_A(k) \\ &= \left(\frac{dT(k)}{dt}\rho C_p + \frac{Q_M(k)}{V} \right) \Big/ \lambda \end{aligned} \tag{10.15}$$

where $\frac{dT(k)}{dt}$ is calculated from measurements using, e.g., an Euler approximation according to $\frac{dT(k)}{dt} \approx \left(T(k+1) - T(k)\right)/t_s$, where t_s is the sampling time. This calculation is of course sensitive to measurement noise.

Having this estimate the differential equation for the concentration in Eq. 10.4 can be solved over one time step to get $\overline{C}_A(k+1)$. Eq. 10.4 can also be rewritten to achieve a linear regression model according to

Eq. 10.7 with

$$\begin{aligned} y(k) &= \log\left(\frac{-\frac{d\overline{C}_A(k)}{dt}\rho C_p}{\alpha\lambda\overline{C}_A(k)}\right) \\ \phi(k) &= \frac{-1}{T(k)+460} \\ \theta &= E/R \end{aligned} \tag{10.16}$$

Here it is seen that this approach assumes that the initial state, i.e., the initial concentration is known.

From simulations it was found that the RLS estimation of the E has converged after around 150 samples. This can be seen in Fig. 10.4, where the result of the RLS estimation with forgetting factor ($\lambda = 0.995$) of E/R for 40 nominal batches together with two batches with an abnormally high and low initial concentration is shown. The nominal batches are simulated using Monte-Carlo simulation with standard deviations of 0.2% for E, 0.5% for C_{A0}, and 2% for h_i. The abnormal batches shown in the figure have a deviation of $\pm 3.2\%$ in C_{A0} and nominal values of E and h_i. The plot shows a large initial transition up to 150 samples and will settle at a constant value if the initial concentration has its nominal value. If instead the initial concentration does not have its nominal value, it makes the estimate drift and the slope (derivative) of the drift is almost directly proportional to the initial concentration. The filtered derivatives, i.e., the slope, of the estimates for the same batches as in Fig. 10.4 can be found in Fig. 10.5. The two abnormal batches clearly shows a larger derivative after 160 samples. Simulations have shown that the derivative of the estimate of E after 200 samples is in fact a very good indicator of C_{A0}. This fact may be used to calculate an estimate of C_A after approximately half the batch by simply integrating Eq. 10.15 from the new estimate of C_{A0}.

The RLS estimation using the measurements from the process is $f(y,u)$ in Fig. 10.3 in this approach.

EKF Approach

The second approach is to use an EKF to estimate C_A even though the true activation energy, E, is not known. The estimate will not be correct but again by using the filtered derivative of the estimated concentration, information about C_{A0} can be extracted. The EKF algorithm in Eq. 10.13

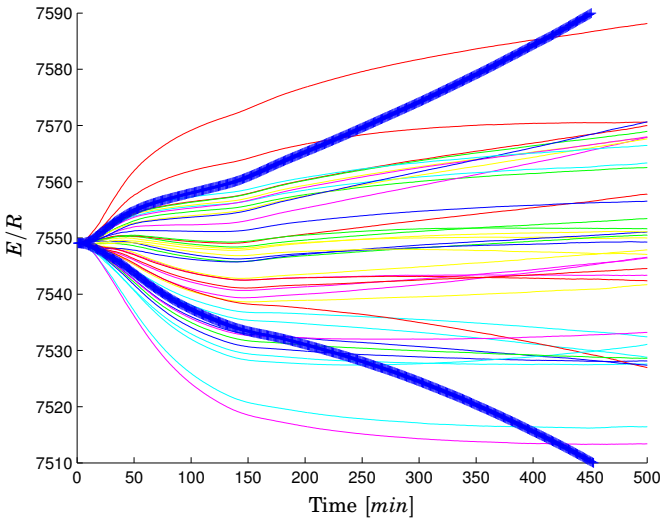

Figure 10.4 RLS estimation with forgetting factor ($\lambda = 0.995$) of E/R for 40 nominal batches (thin lines) together with two batches with an abnormally high and low initial concentration (thick lines).

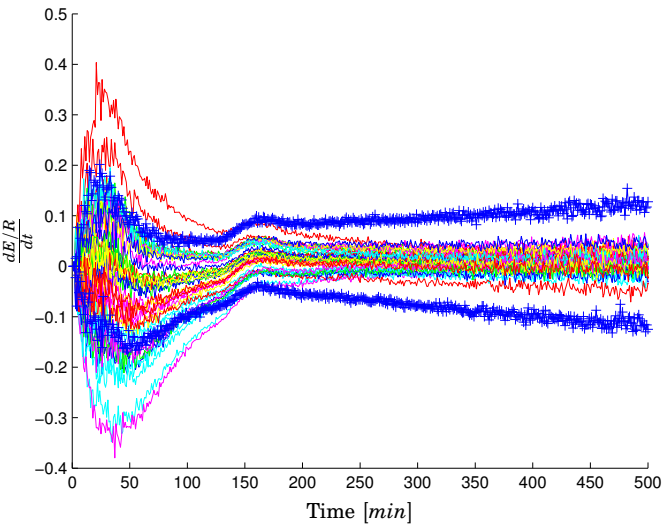

Figure 10.5 Filtered derivative of the RLS estimation of E/R in Fig. 10.4 for 40 nominal batches (thin lines) together with two batches with an abnormally high and low initial concentration (+).

was used with

$$Q = \begin{bmatrix} 10^4 & 0 \\ 0 & 3 \end{bmatrix} \qquad (10.17)$$
$$R = 10^4$$

together with the model described in Eq. 10.4.

The results from this approach is not as intuitive as the results from the RLS approach. The estimation of C_A, with the mean subtracted, can be seen in Fig. 10.6. The filtered derivative of the estimates in Fig. 10.6 can be seen in Fig. 10.7.

Instead of getting a consistent indication of C_{A0} in the filtered derivative of the estimate as in the RLS approach, here the indication is at a specific time interval. The values for the two abnormal batches are significantly outside the normal batches (although this is hard to see in the figure) after about 135 samples and 10 samples forward. The two batches have a deviation twice the value of the largest deviation from the 40 normal batches. This is the kind of behavior in the variables that will lead to that a variable-wise unfolded MPCA method will only signal that the batch is abnormal during this period, while in a batch-wise unfolded MPCA method the batch will be outside the control limits from this time to the end of the batch as discussed in Chapter 9.

The EKF estimation using the measurements from the process is $f(y,u)$ in Fig. 10.3 in this approach.

Comparison using Batch-Vise Unfolded MPCA

The estimate, and its filtered derivative in the two approaches described above are here used as extra variables in addition to the original variables, the temperature in the reactor, T, and the amount of energy added to or withdrawn from the reactor, Q_M, in batch-wise MPCA. This makes the total number of variables $J = 4$. The number of batches $I = 40$ and the number of samples $K = 500$. This gives that the unfolded matrix used in batch-wise MPCA, $X^{I\times KJ} = X^{40\times 2000}$. The projection to model plane method is used for filling in future measurements. As normal variation in the batches for calculating the model C_{A0} has a standard deviation of 0.5% of the nominal value and E has a standard deviation of 0.2% of the nominal value in the Monte-Carlo simulations.

The performance is compared to batch-wise MPCA without the extra variables added for the detection of two different faults. The first fault is a 3.2% high C_{A0} and the second fault is a 1.2% ramp increase of E from 300 to 400 samples. These fault are comparable to the faults used in Chapters 8 and 9.

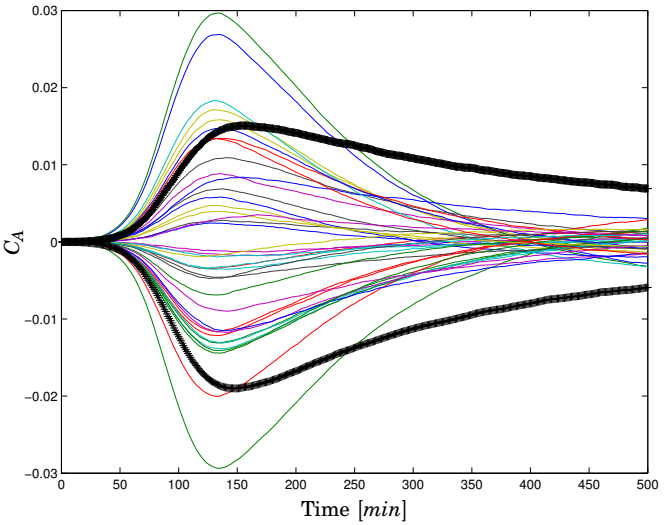

Figure 10.6 EKF estimation of C_A for 40 nominal batches (thin lines) together with two batches with an abnormally high and low initial concentration (thick lines). The mean has been subtracted.

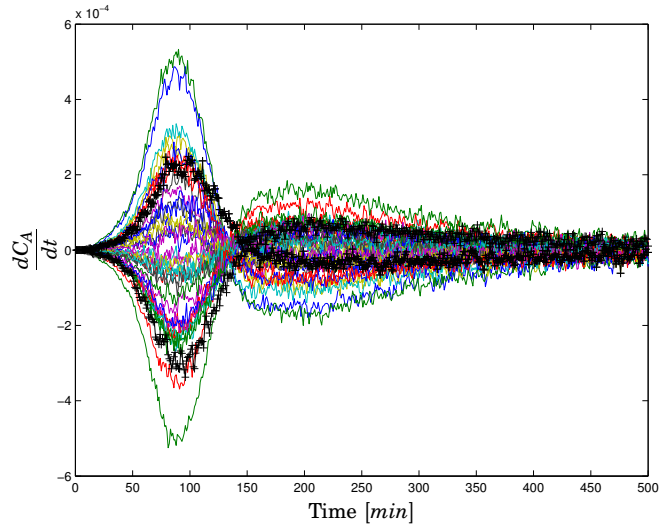

Figure 10.7 Filtered derivative of the EKF estimation of C_A for 40 nominal batches (thin lines) together with two batches with an abnormally high and low initial concentration (+).

Abnormal Deviation in C_{A0} The T^2 index, using three principal components, for the three methods for the fault of 3.2% high C_{A0} can be found in Fig. 10.8. It can be seen that the deviation in C_{A0} is detected at the 99% level after 197 samples for the normal batch-wise MPCA, at 158 samples for the method with RLS estimation of E, and as early as after 132 samples for the method with EKF estimate. The time of detection is decreased by 33% from 197 to 132 samples in the T^2 plot.

In Fig. 10.9 the squared prediction error SPE for the three methods are plotted. Even though it is hard to see, there are three different curves in the plots behaving similarly. The different methods give values of similar magnitude that overlap each other. The fault is not reliably detected in this index, the SPE values jump between being inside and outside the control limits.

It can be concluded that the combination of the model-based estimation and the batch-wise MPCA improves the detection of this fault significantly.

Ramp Increase in E The performance of the detection of a second fault, a 1.2% ramp increase of the activation energy E of from time 300 to 400 is shown in Fig. 10.10, T^2, and Fig 10.11, SPE. In Fig. 10.10 it can be seen that the detection of the fault is somewhat later (10 samples) in the methods where the estimates have been added. This is of minor importance since the fault is detected earlier in the SPE plot, Fig. 10.11, where all the methods have a very similar behavior at the instance of the start of the fault around 300 samples. The three curves overlap almost completely from 300 to 350 samples. The reason that the fault is detected first in the SPE index is described in Chapter 6 in the description of the batch-wise MPCA method.

It can be concluded that all of the methods have similar performance for this fault. Of course, the methods based on estimated variables cannot be anticipated to improve the detection of this fault since it is already detected at the time instance it occurs with the basic method without any extra variables.

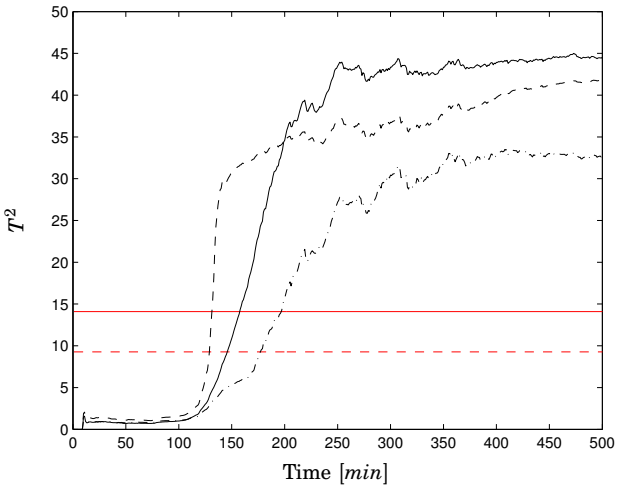

Figure 10.8 T^2 in batch-wise MPCA for 3.2% high C_{A0} in three different variants: Only measured variables T and Q_M (dashed-dotted), T, Q_M, and filtered derivative of the estimate of E using RLS (solid), and T, Q_M, and filtered derivative of the estimate of C_A using EKF (dashed). 95% (dashed) and 99% (solid) confidence limits are also plotted in the figure.

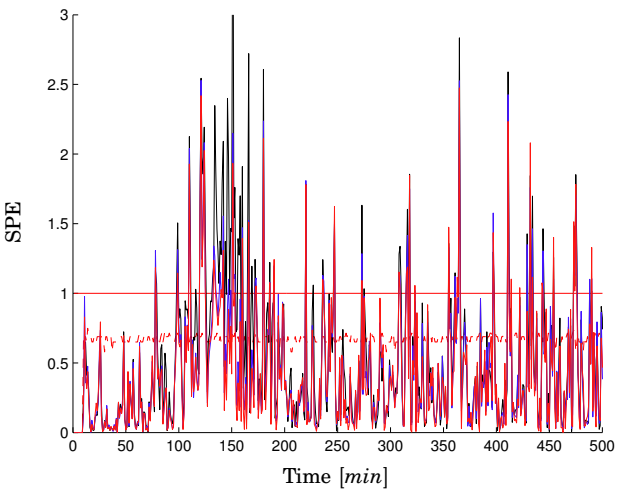

Figure 10.9 SPE in batch-wise MPCA for 3.2% high C_{A0} in three different variants. In the plot the same methods as in Fig. 10.8 are plotted but they overlap with only minor differences. 95% (dashed) and 99% (solid) confidence limits are also plotted in the figure.

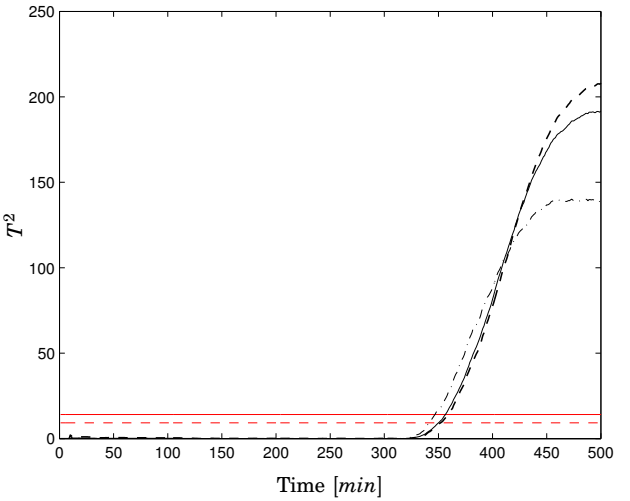

Figure 10.10 T^2 in batch-wise MPCA for a ramp increase of 10% in E from 300 to 400 samples in three different variants: Only measured variables T and Q_M (dashed-dotted), T, Q_M, and filtered derivative of the estimate of E using RLS (solid), and T, Q_M, and filtered derivative of the estimate of C_A using EKF (dashed).

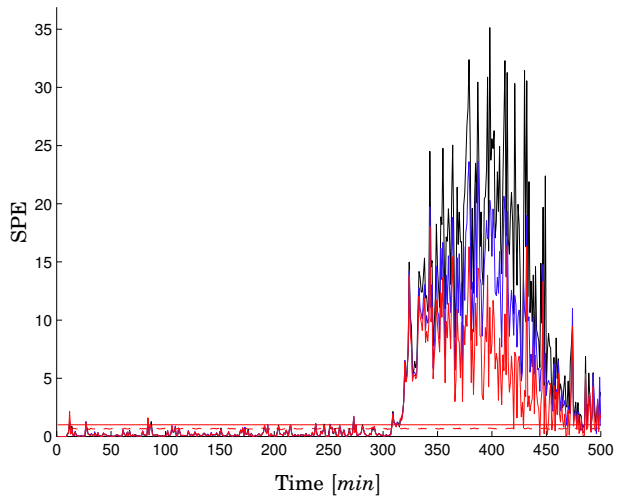

Figure 10.11 SPE in batch-wise MPCA for a ramp increase of 10% in E from 300 to 400 samples in three different variants. In the plot the same methods as in Fig. 10.10 are plotted. At 300 samples all methods detect the fault in a similar way and the curves overlap almost completely from 300 to 350 samples, which is the time interval of interest in this example.

10.5 Summary

In this chapter a new approach on how to combine model-based estimation and multivariate statistical methods for batch process fault diagnosis has been presented and applied in a simulation study on a simplified version of the simulated batch reactor process described in Chapter 7. Previous work in the area has also been described.

The new approach shows significantly better performance for the detection of an initial fault, an abnormally large deviation in C_{A0}, compared to ordinary MPCA, see Fig. 10.8. The approach should also work for the full batch process if the jacket equations are simplified to only using a fluid as described above in the section about simplifications in MBPCA.

Another way of applying the proposed methods is to start the estimation, using RLS or EKF, from the point in time where the cooling phase is started. The estimation would have to start from an initial guess of the state at that time. Of course the proposed methods increase the amount of work needed to implement the diagnosis system. For example, more parameters need to be tuned.

11

Conclusions

The topic of this thesis is fault detection and diagnosis in batch processes both from a control system implementation point of view and from a methodological point of view. In the first part of the thesis a structure for exception handling in recipe-based batch production using JGrafchart has been proposed. In the second part the focus has been on fault detection and isolation in batch processes. Some inherent properties of batch processes have been pointed out that make the diagnosis of a batch process quite different from diagnosis of a continuous process. Several methods from different categories of fault detection approaches have been described and implemented for comparison. A new method to improve fault detection has been proposed. By combining state and parameter estimation with process history-based methods it is possible to improve the detection capability of the history-based methods.

11.1 Exception Handling in Recipe-Based Batch Control

Exception handling is an important area of recipe-based batch control that so far has received little interest from the standardization organizations. Exception handling is a critical element for achieving long-term success in batch production. Correct handling of exceptions is a key element in process safety, consistent product quality, and production cost minimization.

A new approach to equipment supervision in recipe-based batch control systems has been proposed. The proposed method for unit supervision is based on augmenting each equipment object, i.e., unit, equipment module, or control module, with a finite state machine modeling the behavior of the object. The normal execution of a recipe causes the equipment-state machine to change state. In addition to the normal states, error states are added to the state machines. The safety logic in the system is implemented as transitions or guards in the equipment-state machine.

Each operation in the control system is modeled with a procedure-state machine describing the state of the procedure in a similar way as the equipment objects. The operations also use a start-state machine to check if the unit is in a state where the operation is allowed to start. The state machines can be implemented in several ways. In this thesis multiple input multiple output (MIMO) macro steps have been proposed for this. Using the MIMO functionality of the macro step it is possible and convenient to model hierarchical state machines of the proposed type in JGrafchart.

Exception handling is also needed at the recipe level, e.g., an exception that has occurred must be fed back to the control recipe recorded in the batch report, and appropriate actions must be taken. If a batch is aborted the scheduler needs to reschedule the batch for a later time. An important consideration is how to separate the recipe information from the exception handling logic and operations. If the latter is included in the recipe, it becomes difficult to develop, maintain, and use. It has been shown how exception transitions and a new language element, the step fusion sets, can be used for the separation. These new approaches for representing exception handling at the recipe-level gives a clear separation between exception handling logic and the logic for the normal operation.

Different combinations of exception handling at the unit level and the recipe level have been suggested in the thesis. The proposed approaches have been implemented in JGrafchart and tested on a realistic pilot plant, Procel, at UPC in Barcelona, where they have been integrated with recipe execution, resource allocation, and scheduling. The described structures and language elements in JGrafchart should make the development, maintenance, and use of the exception handling logic both at the unit level and at the recipe level an easier task. With well structured and automated exception handling both time and money can be saved. The new structures and concepts described in the thesis also fits nicely into the S88 batch control standard.

Future Research

Here only the basic version of Grafchart has been implemented in JGrafchart. The state machine based structure for modeling equipment objects and operations could be implemented in High-Level Grafchart [Johnsson, 1999]. The state machine would model the behavior of a class, e.g., a certain type of valves, and each token in the state machine would be an instance of the class, e.g. a specific valve, see Fig. 11.1.

To find solutions for different scenarios and plants the proposed approaches should be further tested and implemented in larger batch control systems. The suggested methods need to be tested on industrial plants to determine the practical aspects.

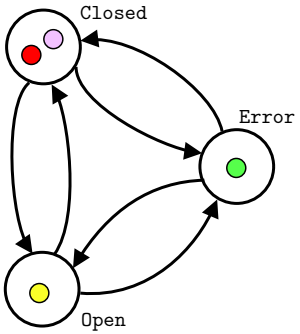

Figure 11.1 An high-level equipment-state machine for valves.

11.2 Batch Process Fault Detection and Isolation

The finite duration and non-linear behavior of batch processes where the variables and the dynamics change significantly over time and the fact that the quality variables are usually only measured at the end of the batch lead to that monitoring of batch processes is quite different from the monitoring of continuous processes. There is also often normal variations between batches, e.g, initial feed variations, ambient temperature, and fouling, which make the process behave differently from batch to batch even though the product is within its specifications. This is often not considered in many of the model-based approaches to fault detection and isolation. Instead a fault is assumed to be a deviation in a single parameter from a specific nominal value. A problem is that when having normal variations in several parameter even a very simple batch process is not fully observable, which leads to that classical estimation techniques, such as extended Kalman filters and recursive least squares, do not give the true values of the states and/or parameters.

Since there will always be differences between batch runs such as initial conditions, parameter changes, disturbances, and measurement noise a successful batch where the end product is within the given specifications does not need to exactly follow the trajectory of the nominal process. This normal batch to batch variation may give false alarms or the control limits need to be raised. If a data base of historically successful batches is available, control limits that give the specified end product can be calculated.

A benchmark batch process simulation model has been implemented to be used for the comparison of different fault detection methods. The simulation model makes it possible to simulate normal behavior of the process to get historical data, as well as to simulate how the process is

influenced by different faults, i.e., to generate fault signatures. A dynamic model gives a deep knowledge about the process and can be used at different stages for designing both the control and the diagnostic system.

The diagnostic model processor, DMP, method and the deep model algorithm, DMA, method, two model-based fault detection method developed for continuous processes, have been applied to the benchmark simulated batch process. Both of the methods rewrite the dynamic models of the system to form residuals. The residuals are used to detect faults by assuming that when the process is behaving normally the residuals should be small and when a fault occurs they will become large since the model is no longer valid. Since there are disturbances and measurement noise the residuals will not be exactly equal to zero and tolerances and sensitivities need to be determined for each residual.

For a batch process the tolerances and the sensitivities vary over time and need to be determined for different time intervals or phases in the process. It has also been shown that the methods are sensitive to noise and that the residuals need to be chosen with care to achieve good performance. Some other shortcomings of the methods have also been pointed out. Both the methods assume that only one fault is present at the same time and the rest of the system is acting at its nominal values. It has been shown that under this assumption the two methods can detect single faults, but not all faults can be isolated from each other.

Multivariate statistical methods take normal batch to batch variations into consideration and only historical data from successful batch runs is used to model the normal behavior of the process. Statistical control limits from the normal batches are derived and future batches are compared to these limits to classify the batches as normal or abnormal.

The multivariate statistical methods compared in the thesis are based on principal component analysis, PCA. PCA tries to explain the covariance structure of a set of variables by finding a small number of linear combinations of the variables. PCA linearly transforms an original set of correlated variables into an often substantially smaller set of uncorrelated variables, called principal components, that represent most of the information in the original set of variables. The monitoring of the process then takes place in the space of the uncorrelated variables, instead of using the original variables, using statistical control limits.

A survey and comparison of different fault detection methods has been performed using the simulated batch process. In the thesis it has been pointed out how the different methods are related to each other and when they are equal. It has been shown how different unfoldings of three-dimensional data from a batch process affect the control limits and the sensitivity to noise has been pointed out for the moving window PCA method. Some shortcomings of the methods are discussed and new

algorithms have been developed for the methods BDPCA and DPARAFAC.

In the thesis some of the attempts to combine different methods for fault diagnosis have been discussed. Especially the MBPCA method has been described together with some of its shortcomings. A new approach for combining model-based estimation and multivariate statistical methods for batch process monitoring has been proposed. The estimation methods recursive least squares, RLS, and extended Kalman filtering, EKF, have been combined with the multivariate statistical method batch-wise unfolded MPCA. Even though the estimation does not give the correct values of the estimated states or parameters, the estimation and its time derivative still contain information about the process which can be compared to the same estimations for the normal batches in the historical data base. It has been shown that this combination can improve the detection of batches with an initial fault.

Future Research

It has been shown that a dynamic model can be very useful in combination with multivariate statistical methods. In the thesis a simulated process is used. This has given great insight in the problematic parts of batch process monitoring and it has enabled changes to be made to test ideas as the project has progressed.

The work on combining model-based estimation and multivariate statistical method should be further developed and tested on an industrial batch process. The influence of model uncertainty should be looked into. Since the same estimation method is used for both the historical data from normal batches and the new batch monitored online the need for perfect accuracy may not be necessary. It might be sufficient to use a dynamic model describing only a part of the process to improve the monitoring results.

The work in this thesis has been focused mainly on detection and not as much on isolation. Further research is needed in the area of fault isolation in multivariate statistical methods. A dynamic model can be used to simulate different faults to obtain their signatures in the scores, T^2, and the SPE. The fault signature can then be used to isolate the origin of the fault. The work on fault signatures in [Yoon and MacGregor, 2001a] is promising and could be further developed. The work in this thesis has mainly been focused on batch process fault detection of single faults. Isolation of faults becomes even harder when multiple faults occurring during the same time are considered, since the faults interact with each other.

A

Higher-Order Singular Value Decomposition

In [De Lathauwer *et al.*, 2000a] a method for singular value decomposition for higher-order tensors is proposed, called higher-order singular value decomposition, HOSVD. The method is a generalization of SVD for matrices, see Def. 6.2. The article uses tensor notation and the aim in this part of the thesis is to compare the work on HOSVD, and also to some extent translate it, to the notation used for the multivariate statistical methods in Chapter 6.

First a notation for the unfolding of a tensor is defined.

DEFINITION A.1—UNFOLDING OF A TENSOR
The tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ unfolded along its nth mode is denoted

$$A_{(n)} \in \mathbb{R}^{I_n \times I_{n+1} \ldots I_N I_1 \ldots I_{n-1}}$$

□

For example, a third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ unfolded in the three modes becomes

$$\begin{aligned} X_{(1)} &= X^{I \times JK} \\ X_{(2)} &= X^{J \times KI} \\ X_{(3)} &= X^{K \times IJ} \end{aligned} \tag{A.1}$$

$X_{(1)}$ and $X_{(2)}$ are equivalent to the two unfoldings in Fig. 6.4 after rearranging rows and columns.

To be able to compare HOSVD with the ordinary SVD for matrices a number of definitions will be introduced.

The rank of a higher-order tensor cannot be as easily defined as the rank of a matrix. The rank of a matrix is the number of linearly independent rows or columns of the matrix, or the number of nonzero singular values.

DEFINITION A.2—RANK-1 TENSOR
The tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ has rank $R = 1$ when it equals the outer product of N vectors, $U^{(1)}, U^{(2)}, \ldots, U^{(N)}$, i.e., when each element in $\underline{\mathbf{A}}$, $a_{i_1 i_2 \ldots i_N}$, can be written as

$$a_{i_1 i_2 \ldots i_N} = u^{(1)}_{i_1} u^{(2)}_{i_2} \ldots u^{(N)}_{i_N}$$

where $u^{(n)}_{i_n}$ is the i_nth element in the nth vector. This can also be written, by using $\wedge$ for the outer product, as

$$\underline{\mathbf{A}} = U^{(1)} \wedge U^{(2)} \wedge \ldots \wedge U^{(N)}.$$

□

DEFINITION A.3—RANK OF A TENSOR
The rank of a tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is the minimal number of rank-1 tensors that give $\underline{\mathbf{A}}$ in a linear combination. The rank of a tensor is denoted

$$R = \text{rank}\,(\underline{\mathbf{A}})$$

□

This is in analogy with the decomposition of a rank-R matrix as a sum of rank-1 terms.

If the *n-mode vectors* of $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ are defined as the I_n-dimensional vectors obtained by varying index i_n and keeping all the other indices fixed, then the n-rank of the tensor can be defined as in Def. A.4.

DEFINITION A.4—n-RANK OF A TENSOR
The n-rank of a tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, denoted by

$$R_n = \text{rank}_n\,(\underline{\mathbf{A}})\,,$$

is the dimension of the vector space spanned by the n-mode vectors. □

If $\underline{\mathbf{A}}$ is unfolded to the matrix $A_{(n)}$ using Def. A.1, it can be seen that the n-mode vectors of $\underline{\mathbf{A}}$ are the column vectors of A_n and

$$\text{rank}_n\,(\underline{\mathbf{A}}) = \text{rank}\left(A_{(n)}\right).$$

This is a generalization of the column (row) rank of matrices. Note that the different n-ranks of a tensor do not have to be the same and the rank of a tensor does not have to be equal to an n-rank, not even if the n-ranks are the same. From the Def. A.4 and A.3 it can be seen that $R_n \leq R$.

DEFINITION A.5—SCALAR PRODUCT OF TWO TENSORS
The scalar product of two tensors $\underline{\mathbf{A}}, \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is defined as

$$\langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} b_{i_1 i_2 \ldots i_N} a_{i_1 i_2 \ldots i_N}$$

□

DEFINITION A.6—FROBENIUS NORM
The Frobenius norm of a tensor $\underline{\mathbf{A}}$ is defined as

$$\|\underline{\mathbf{A}}\|_F = \sqrt{\langle \underline{\mathbf{A}}, \underline{\mathbf{A}} \rangle}$$

□

DEFINITION A.7—ORTHOGONALITY
Two tensors $\underline{\mathbf{A}}, \underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ are orthogonal if

$$\langle \underline{\mathbf{A}}, \underline{\mathbf{B}} \rangle = 0$$

□

DEFINITION A.8—MATRIX-TENSOR PRODUCT
The product of a tensor $\underline{\mathbf{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ and a matrix $H \in \mathbb{R}^{J_n \times I_n}$, where $n = 1, 2, \ldots, N$, is defined as

$$\underline{\mathbf{B}} = \underline{\mathbf{A}} \times_n H \tag{A.2}$$

where $\underline{\mathbf{B}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_{n-1} \times J_n \times I_{n+1} \times \ldots \times I_N}$ and the entries in $\underline{\mathbf{B}}$ are given by

$$\underline{\mathbf{B}}_{i_1 i_2 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_N} a_{i_1 i_2 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} H_{j_n i_n} \tag{A.3}$$

□

For example the product of $\underline{\mathbf{X}} \in \mathbb{R}^{I \times J \times K}$ and $H \in \mathbb{R}^{L \times I_n}$, where $n = 1, 2, 3$ and thus $I_1 = I$, $I_2 = J$, and $I_3 = K$, and $\underline{\mathbf{B}}$ will have the dimensions

$$\begin{aligned} \underline{\mathbf{B}} &= \underline{\mathbf{X}} \times_1 H \in \mathbb{R}^{L \times J \times K}, \ n = 1 \\ \underline{\mathbf{B}} &= \underline{\mathbf{X}} \times_2 H \in \mathbb{R}^{I \times L \times K}, \ n = 2 \\ \underline{\mathbf{B}} &= \underline{\mathbf{X}} \times_3 H \in \mathbb{R}^{I \times J \times L}, \ n = 3 \end{aligned} \tag{A.4}$$

The tensors in Eq. A.4 can after unfolding also be written as

$$\begin{aligned} B_{(1)} &= H \cdot A_{(1)}, \ n = 1 \\ B_{(2)} &= H \cdot A_{(2)}, \ n = 2 \\ B_{(3)} &= H \cdot A_{(3)}, \ n = 3 \end{aligned} \tag{A.5}$$

where $B_{(n)}$ can be refolded to its corresponding tensor, $\underline{\mathrm{B}}$.

The tensor matrix product in Def. A.8 have the following properties [De Lathauwer *et al.*, 2000a]

1. If $F \in \mathbb{R}^{J_n \times I_n}$ and $H \in \mathbb{R}^{J_m \times I_m}$ and $n \neq m$ then

$$(\underline{\mathrm{A}} \times_n F) \times_m H = (\underline{\mathrm{A}} \times_m H) \times_n F = \underline{\mathrm{A}} \times_n F \times_m H$$

2. If $F \in \mathbb{R}^{J_n \times I_n}$ and $G \in \mathbb{R}^{M \times J_n}$ then

$$(\underline{\mathrm{A}} \times_n F) \times_n H = \underline{\mathrm{A}} \times_n (H \cdot F)$$

THEOREM A.1—HIGHER ORDER SINGULAR VALUE DECOMPOSITION – HOSVD [DE LATHAUWER *et al.*, 2000A]
The Nth-order tensor $\underline{\mathrm{A}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ can be decomposed according to

$$\underline{\mathrm{A}} = \underline{\mathrm{S}} \times_1 U^{(1)} \times_2 U^{(2)} \ldots \times_N U^{(N)} \tag{A.6}$$

where $U^{(n)} \in \mathbb{R}^{I_n \times I_n}$ is a unitary matrix such that $\left(U^{(n)}\right)\left(U^{(n)}\right)^T = I^{(n)}$, for $n = 1, 2, \ldots, N$, and $\underline{\mathrm{S}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$.

The subtensors of $\underline{\mathrm{S}}$, $\underline{\mathrm{S}}_{i_n=\alpha}$, are the matrices constructed from slices of $\underline{\mathrm{S}}$ by setting the index of one of the modes, i.e., $n = 1, 2, \ldots, N$, equal to α. The subtensors have the following two properties.

1. Two subtensors fixed in the same mode are orthogonal, i.e.,

$$\langle \underline{\mathrm{S}}_{i_n=\alpha}, \underline{\mathrm{S}}_{i_n=\beta} \rangle = 0, \ \alpha \neq \beta$$

2. The norms of the subtensors are ordered according to

$$\|\underline{\mathrm{S}}_{i_n=1}\|_F \geq \|\underline{\mathrm{S}}_{i_n=2}\|_F \geq \ldots \geq \|\underline{\mathrm{S}}_{i_n=I_n}\|_F \geq 0$$

□

For the proof see [De Lathauwer *et al.*, 2000a].

In analogy with the singular values and the singular vectors for matrices in Def. 6.2 the Frobenius norm of the sub-tensor, $\|\underline{\mathrm{S}}_{i_n=i}\|_F = \sigma_i^{(n)}$, is the ith n-mode singular value and $U_i^{(n)}$ is the ith n-mode singular vector.

It can be noted that the model described in Eq. A.6 has the same structure as the Tucker3 model in Fig. 6.5 with $\underline{\mathrm{G}} = \underline{\mathrm{S}}, A = U^{(1)}, B = U^{(2)}, C = U^{(3)}, D = I, E = J$, and $F = K$.

By using the Kronecker tensor product in Def. 6.3, Eq. A.6 can be written as

$$A_{(n)} = U^{(n)} S_{(n)} \left(U^{(n+1)} \otimes \ldots \otimes U^{(N)} \otimes U^{(1)} \otimes \ldots \otimes U^{(n-1)} \right)^T \tag{A.7}$$

and using SVD on the unfoldings gives

$$A_{(n)} = U^{(n)} \Sigma^{(n)} (V^{(n)})^T \tag{A.8}$$

for $n = 1, 2, \ldots, N$.

As an example applying these results on the third-order tensor $\underline{\mathrm{X}} \in \mathbb{R}^{I \times J \times K}$ gives

$$\begin{aligned} X_{(1)} &= U^{(1)} S_{(1)} \left(U^{(2)} \otimes U^{(3)} \right)^T = X^{I \times JK};\ S_{(1)} = S^{I \times JK} \\ X_{(2)} &= U^{(2)} S_{(2)} \left(U^{(3)} \otimes U^{(1)} \right)^T = X^{J \times KI};\ S_{(2)} = S^{J \times KI} \\ X_{(3)} &= U^{(3)} S_{(3)} \left(U^{(1)} \otimes U^{(2)} \right)^T = X^{K \times IJ};\ S_{(3)} = S^{K \times IJ} \end{aligned} \tag{A.9}$$

and by using SVD these unfoldings can be written as

$$\begin{aligned} X^{I \times JK} &= U^{(1)} \Sigma^{(1)} (V^{(1)})^T \\ X^{J \times KI} &= U^{(2)} \Sigma^{(2)} (V^{(2)})^T \\ X^{K \times IJ} &= U^{(3)} \Sigma^{(3)} (V^{(3)})^T \end{aligned} \tag{A.10}$$

The main difference between the matrix SVD and the HOSVD is that $\underline{\mathrm{S}}$ is a full tensor and does not have a diagonal structure as in the matrix version. This means that one does not get an optimal approximation of $\underline{\mathrm{X}}$ by choosing the first D, E, and F singular vectors in $U^{(1)}, U^{(2)}$, and $U^{(3)}$ respectively, which actually is the original approach in the work on Tucker3.

In [De Lathauwer *et al.*, 2000b] the work on HOSVD is continued and algorithms for the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximations of higher-order tensors are developed. The best rank-1 approximation equals calculating a PARAFAC model with only one component, i.e., $R = 1$, and the best rank-$(R_1, R_2, \ldots, R_N)$ approximation is the same as calculating a Tucker3 model with dimensions (R_1, R_2, R_3) when $N = 3$. The algorithm for rank-1 approximation has the same problem as the

PARAFAC algorithm that the solution may only be a local minimum. As a starting point of the algorithm it is suggested that the HOSVD solution truncated after the first term is used.

The algorithm proposed for the best rank-$(R_1, R_2, \ldots, R_N)$ approximation is a square-root version of the algorithm developed by Kroonenberg for Tucker3 [Kroonenberg, 1983] and it can be compared to the SVD method developed in [Andersson and Bro, 1998].

B

Schedule in BatchML format

```
<BatchListEntry ID="66">
    <BatchListEntryType>Batch</BatchListEntryType>
    <Status>Scheduled</Status>
    <RecipeID>7</RecipeID>
    <BatchID>66</BatchID>
    <ProductID>1</ProductID>
    <OrderID>No Order</OrderID>
    <BatchPriority>No Order</BatchPriority>
    <RequestedBatchSize>1 Batch</RequestedBatchSize>
    <ActualBatchSize />
    <BatchListEntry ID="9">
        <BatchListEntryType>UnitProcedure</BatchListEntryType>
        <Status>Scheduled</Status>
        <EquipmentID>5</EquipmentID>
        <ActualEquipmentID />
        <BatchListEntry ID="11">
            <BatchListEntryType>Operation</BatchListEntryType>
            <Status>Scheduled</Status>
            <RequestedStartTime>0</RequestedStartTime>
            <RequestedEndTime>2.63</RequestedEndTime>
            <ActualStartTime />
            <ActualEndTime />
        </BatchListEntry>
        <BatchListEntry ID="13">
            <BatchListEntryType>Operation</BatchListEntryType>
            <Status>Scheduled</Status>
            <RequestedStartTime>2.63</RequestedStartTime>
            <RequestedEndTime>3.63</RequestedEndTime>
            <ActualStartTime />
            <ActualEndTime />
```

```
        </BatchListEntry>
        <BatchListEntry ID="15">
            <BatchListEntryType>Operation</BatchListEntryType>
            <Status>Scheduled</Status>
            <RequestedStartTime>3.63</RequestedStartTime>
            <RequestedEndTime>5.63</RequestedEndTime>
            <ActualStartTime />
            <ActualEndTime />
        </BatchListEntry>
    </BatchListEntry>
    <BatchListEntry ID="17">
        <BatchListEntryType>UnitProcedure</BatchListEntryType>
        <Status>Scheduled</Status>
        <EquipmentID>6</EquipmentID>
        <ActualEquipmentID />
        <BatchListEntry ID="19">
            <BatchListEntryType>Operation</BatchListEntryType>
            <Status>Scheduled</Status>
            <RequestedStartTime>3.63</RequestedStartTime>
            <RequestedEndTime>5.63</RequestedEndTime>
            <ActualStartTime />
            <ActualEndTime />
        </BatchListEntry>
    </BatchListEntry>
    <BatchListEntry ID="21">
        <BatchListEntryType>UnitProcedure</BatchListEntryType>
        <Status>Scheduled</Status>
        <EquipmentID>3</EquipmentID>
        <ActualEquipmentID />
        <BatchListEntry ID="24">
            <BatchListEntryType>Operation</BatchListEntryType>
            <Status>Scheduled</Status>
            <RequestedStartTime>3.63</RequestedStartTime>
            <RequestedEndTime>5.63</RequestedEndTime>
            <ActualStartTime />
            <ActualEndTime />
        </BatchListEntry>
        <BatchListEntry ID="26">
            <BatchListEntryType>Operation</BatchListEntryType>
            <Status>Scheduled</Status>
            <RequestedStartTime>5.63</RequestedStartTime>
            <RequestedEndTime>8.08</RequestedEndTime>
            <ActualStartTime />
```

```
            <ActualEndTime />
        </BatchListEntry>
        <BatchListEntry ID="28">
            <BatchListEntryType>Operation</BatchListEntryType>
            <Status>Scheduled</Status>
            <RequestedStartTime>8.08</RequestedStartTime>
            <RequestedEndTime>12.5</RequestedEndTime>
            <ActualStartTime />
            <ActualEndTime />
        </BatchListEntry>
        <BatchListEntry ID="30">
            <BatchListEntryType>Operation</BatchListEntryType>
            <Status>Scheduled</Status>
            <RequestedStartTime>12.5</RequestedStartTime>
            <RequestedEndTime>14</RequestedEndTime>
            <ActualStartTime />
            <ActualEndTime />
        </BatchListEntry>
    </BatchListEntry>
</BatchListEntry>
```

References

Åkesson, M. (1997): "Integrated control and fault detection for a mechanical servo process." In *Proceedings of IFAC Safeprocess '97*, pp. 1252–1257.

American Institute of Chemical Engineers – AIChE (1999): *Guidelines for Process Safety in Batch Reaction Systems*. AIChE.

Andersson, C. A. and R. Bro (1998): "Improving the speed of multi-way algorithms: Part1. Tucker3." *Chemometrics and Intelligent Laboratory Systems*, **42**, pp. 93–103.

Andersson, C. A. and R. Bro (2000): "The N-way Toolbox for MATLAB." *Chemometrics and Intelligent Laboratory Systems*, **52:1**, pp. 1–4. http://www.models.kvl.dk/source/nwaytoolbox/.

ANSI/ISA (1995): "ANSI/ISA 88.01 – Batch Control Part 1: Models and Terminology." The Instrumentation, Systems, and Automation Society.

ANSI/ISA (2001): "ANSI/ISA 88.00.02 – Batch Control Part 2: Data Structures and Guidelines for Languages." The Instrumentation, Systems, and Automation Society.

ANSI/ISA (2003): "ANSI/ISA 88.00.03 – Batch Control Part 3: General and Site Recipe Models and Representation." The Instrumentation, Systems, and Automation Society.

ANSI/ISA (2004): "ANSI/ISA 84.00.01 Safety Instrumented Systems for the Process Industry Sector – Part 1: Framework, Definitions, System, Hardware and Software Requirements." The Instrumentation, Systems, and Automation Society.

Arbiza, M., J. Cantón, A. Espuña, and L. Puigjaner (2003a): "Flexible rescheduling tool for short-term plan updating." In *Proc. of AIChE'03 Annual Meeting*.

Arbiza, M., J. Cantón, A. Espuña, and L. Puigjaner (2003b): "Objective based schedule selector: a rescheduling tool for short-term plan updating." In *Proc. of 14th European Symposium on Computer Aided Process Engineering (ESCAPE 14)*.

Arteaga, F. and A. Ferrer (2002): "Dealing with missing data in MSPC: several methods, different interpretations, some examples." *Journal of Chemometrics*, **16**, pp. 408–418.

Årzén, K.-E. (1991): "Sequential function charts for knowledge-based, real-time applications." In *Proc. Third IFAC Workshop on AI in Real-Time Control*. Rohnert Park, California.

Årzén, K.-E. (1993): "Grafcet for intelligent real-time systems." In *Preprints IFAC 12th World Congress*. Sydney, Australia.

Årzén, K.-E. (1994): "Grafcet for intelligent supervisory control applications." *Automatica*, **30:10**, pp. 1513–1526.

Årzén, K.-E., R. Olsson, and J. Åkesson (2002): "Grafchart for Procedural Operator Support Tasks." In *Proceedings of the 15th IFAC World Congress, Barcelona, Spain*.

Åström, K. J. and B. Wittenmark (1995): *Adaptive Control*. Addison-Wesley, Reading, Massachusetts.

Åström, K. J. and B. Wittenmark (1997): *Computer-Controlled Systems*. Prentice Hall, Upper Saddle River, New Jersey.

Bonfill, A., M. Arbiza, E. Musulin, A. Espuña, and L. Puigjaner (2004): "Integrating robustness and fault diagnosis in on-line scheduling of batch chemical plants." In *Proc. of International IMS Forum*, pp. 515–522.

Bonné, D. and S. B. Jørgensen (2004): "Data-driven modeling of batch processes." In *Proc. of 7th International Symposium on Advanced Control of Chemical Processes, ADCHEM*.

Brettschmeider, H., H. Genrich, and H. Hanisch (1996): "Verification and performance analysis of recipe based controllers by means of dynamic plant models." In *Second International Conference on Computer Integrated Manufacturing in the Process Industries*. Eindhoven, The Netherlands.

Bro, R. (1997): "PARAFAC. Tutorial and applications." *Chemometrics and Intelligent Laboratory Systems*, **38**, pp. 149–171.

Bro, R. (1998): *Multi-way Analysis in the Food Industry*. PhD thesis, Chemometrics Group, Food Technology, Dept. of Dairy and Food Sciences, Royal Veterinary and Agricultural University, Denmark.

Bro, R. and A. K. Smilde (2003): "Centering and scaling in component analysis." *Journal of Chemometrics*, **17**, pp. 16–33.

Cantón, J., D. Ruiz, C. Benqlilou, J. Nougués, and L. Puigjaner (1999): "Integrated information system for monitoring, scheduling and control applied to batch chemical processes." In *Proc. of the 7th IEEE Int. Conf. on Emerging Technologies and Factory Automation*.

Cantón Padilla, J. (2003): "Intgrated Support System for Planning and Scheduling of Batch Processes." Ph.D. thesis. Department d'Enginyeria Quimica, Universitat Politécnica de Catalunya, Barcelona, Spain.

Carroll, J. D. and J. Chang (1970): "Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition." *Psychometrika*, **35**, pp. 283–319.

Chang, I.-C., C.-C. Yu, and C.-T. Liou (1994): "Model-Based Approach for Fault Diagnosis. 1. Principles of Deep Model Algorithm." *Ind. Eng. Chem. Res.*, **33**, pp. 1542–1555.

Chen, J. (2004): Personal correspondence.

Chen, J. and J. Liu (2001): "Derivation of function space analysis based PCA control charts for batch process monitoring." *Chemical Engineering Science*, **56**, pp. 3289–3304.

Chen, J. and K.-C. Liu (2002): "On-line batch process monitoring using dynamic PCA and dynamic PLS models." *Chemical Engineering Science*, **57**, pp. 63–75.

Chen, J. and J.-H. Yen (2003): "Three-Way Data Analysis with Time Lagged Window for On-Line Batch Process Monitoring." *Korean Journal of Chemical Engineering*, **20:6**, pp. 1000–1011.

Chen, Y., J. F. MacGregor, and T. Kourti (2002): "Multi-way PCA Approaches for the Analysis and Monitoring of Batch Processes: A Critical Assessment." In *AIChE Annual Meeting*.

Choi, S. W., C. Lee, J.-M. Lee, J. H. Park, and I.-B. Lee (2005): "Fault detection and identification of non-linear processes based on kernel PCA." *Chemometrics and intelligent laboratory systems*, **75**, pp. 55–67.

Christie, D. (1998): "A Methodology for Batch Control Implementation – A Real World Lesson." In *Proc. of World Batch Forum – Meeting of the Minds*.

Cinar, A. and C. Undey (1999): "Statistical process and controller performance monitoring. a tutorial on current methods and future directions." In *Proc. of the American Control Conference*, pp. 2625–2639.

Dahl, K. S., M. J. Piovoso, and K. A. Kosanovich (1999): "Translating third-order data analysis methods to chemical batch processes." *Chemometrics and Intelligent Laboratory Systems*, **46:2**, pp. 161–180.

David, R. and H. Alla (1992): *Petri Nets and Grafcet: Tools for modelling discrete events systems*. Prentice-Hall International (UK) Ltd.

De Lathauwer, L., B. De Moor, and J. Vandewalle (2000a): "A multilinear singular value decomposition." *SIAM Journal on Matrix Analysis and Applications*, **21:4**, pp. 1253–1278.

De Lathauwer, L., B. De Moor, and J. Vandewalle (2000b): "On the Best Rank-1 and Rank(R_1, R_2, ..., R_N) Approximation of Higher-Order Tensors." *SIAM Journal on Matrix Analysis and Applications*, **21:4**, pp. 1324–1342.

Dochain, D. (2003): "State and parameter estimation in chemical and biochemical processes: A tutorial." *Journal of Process Control*, **13**, pp. 801–818.

Dong, D. and T. J. McAvoy (1995): "Multi-stage batch process monitoring." *Proc. of American Control Conference*, pp. 1857–186.

Fickelscherer, R. J. (1990): *Automated Process Fault Analysis*. PhD thesis, University of Delaware.

Fisher, T. G. (1990): *Batch Control Systems: Design, Application, and Implementation*. Instrument Society of America, Research Park, NC.

Frank, P. M. (1990): "Fault Diagnosis in Dynamic Systems Using Analytical and Knowledge-based Redundancy – A Survey and Some New Results." *Automatica*, **26:3**, pp. 459–474.

Frisk, E. (2001): *Residual Generation for Fault Diagnosis*. PhD thesis, Linköping University.

Fritz, M., A. Liefeldt, and S. Engell (1999): "Recipe-driven batch processes: Event handling in hybrid system simulation." In *Proc. of the 1999 IEEE International Symposium on Computer Aided Control System Design*.

García-Muñoz, S., T. Kourti, and J. F. MacGregor (2003): "Model Predictive Monitoring for Batch Processes." *Industrial & Engineering Chemistry Research*, **43**, pp. 5929–5941.

Genrich, H. J., H.-M. Hanisch, and K. Wöllhaf (1994): "Verification of recipe-based control procedures by means of predicate/transition nets." In *15th International Conference on Application and Theory of Petri nets, Zaragoza, Spain*.

Gensym Corporation (1995): *G2 Reference Manual*. Gensym Corporation, 125 Cambridge Park Drive, Cambridge, MA 02140, USA.

Gertler, J. (1998): *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, New York, NY.

Gertler, J., W. Li, Y. Huang, and T. McAvoy (1999): "Isolation Enhanced Principal Component Analysis." *AIChE Journal*, **45**, pp. 323–334.

Gertler, J. and T. J. McAvoy (1997): "Principal Component Analysis and Parity Relations – A Strong Duality." In *IFAC SAFEPROCESS'97*, pp. 837–842.

Gertler, J. and D. Singer (1990): "A new structural framework for parity equation-based failure detection and isolation." *Automatica*, **26**, pp. 381–388.

Gertler, J. J. and M. M. Kunwer (1995): "Optimal residual decoupling for robust fault diagnosis." *International Journal of Control*, **61:2**, pp. 395–421.

Golub, G. H. and C. F. Van Loan (1996): *Matrix Computations. 3rd Edition*. The John Hopkins University Press, Baltimore, Maryland.

Graells, M., J. Cantón, B. Peschaud, and L. Puigjaner (1998): "General approach and tool for the scheduling of complex production systems." *Computers and Chemical Engineering*, **22**, pp. 395–402.

Gregersen, L. (2004): *Monitoring and Fault Diagnosis of Fermentation Processes*. PhD thesis, Technical University of Denmark, DTU.

Gregersen, L. and S. B. Jørgensen (1999): "Supervision of fed-batch fermentations." *Chemical Engineering Journal*, **75**, pp. 69–76.

Gurden, S. P., J. A. Westerhuis, S. Bijlsma, and A. K. Smilde (2001): "Modelling of spectroscopic batch process data using grey models to incorporate external information." *Journal of Chemometrics*, **15**, pp. 101–121.

Hanisch, H.-M. and S. Fleck (1996): "A resource allocation scheme for flexible batch plants based on high-level petri nets." In *Proc. of CESA96 IMACS Multiconference*.

Harel, D. (1987): "Statecharts, a visual formalism for complex systems." *Science of Computer Programmming*, **8:3**, pp. 231–274.

Harshman, R. A. (1970): "Foundation of the PARAFAC procedure: Model and conditions for an 'explanatory' multi-mode factor analysis." *UCLA Working Papers in Phonetics*, **16**, pp. 1–84.

Harshman, R. A. and M. E. Lundy (1984): "The PARAFAC model for three-way factor analysis and multidimensional scaling." In *Research methods for multimode data analysis, G. Law, C. W. Snyder, Jr., J. Hattie, and R. P. McDonald (Eds.)*, pp. 122–215. Praeger, New York.

Hoerl, A. E. and R. W. Kennard (1970): "Ridge regression: Biased estimation for nonorthogonal problems." *Technometrics*, **12**, pp. 55–67.

Hotelling, H. (1933): "Analysis of a complex of statistical variables into principal components." *J. of Educational Psychology*, **24**, pp. 417–441, 498–520.

Hyvärinen, A. and E. Oja (2000): "Independent component analysis: Algorithms and applications." *Neural Networks*, **13**, pp. 411–430.

IEC (1985): "IEC 60812 – Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)." International Electrotechnical Commission.

IEC (1990): "IEC 61025 – Fault tree analysis (FTA)." International Electrotechnical Commission.

IEC (1993): "IEC 61131-3 – Programmable Controllers Part 3: Programming Languages."

IEC (1997): "IEC 61512-1 – Batch Control Part 1: Models and Termonology."

IEC (1998): "IEC 61508 – Functional safety of electrical/electronic/programmable electronic safety-related systems."

IEC (2001a): "IEC 61512-2 – Batch Control Part 2: Data Structures and Guidelines for Languages."

IEC (2001b): "IEC 61882 – Hazard and operability studies (HAZOP studies) – Application guide." International Electrotechnical Commission.

IEC (2003): "IEC 61511-1 – Functional safety – Safety instrumented systems for the process industry sector."

Iri, M., K. Aoki, E. O'Shima, and H. Matsyama (1979): "An algorithm for diagnosis of system failures in the chemical process." *Computers & Chemical Engineering*, **3**, pp. 489–493.

ISA (1995): "ISA S88.01 Batch Control." Instrument Society of America.

Isermann, R. (1984): "Process Fault Detection Based on Modeling and Estimation Methods – A Survey." *Automatica*, **20:4**, pp. 387–404.

Isermann, R. (1989): "Process fault diagnosis based on dynamic models and parameter estimation methods." In Patton *et al.*, Eds., *Fault Diagnosis in Dynamic Systems – Theory and Application.*, pp. 253–291. Prentice Hall International.

Isermann, R. (1993): "Fault Diagnosis of Machines via Parameter Estimation and Knowledge Processing – Tutorial Paper." *Automatica*, **29:4**, pp. 815–835.

Isidori, A. (1995): *Nonlinear Control Systems*. Springer-Verlag, Berlin and Heidelberg, Germany.

Isidori, A., M. Kinnaert, V. Cocquempot, C. D. Persis, P. M. Frank, and D. N. Shields (2001): "Residual Generation for FDI in Non-linear Systems." In Åström *et al.*, Eds., *Control of Complex Sytems*, pp. 209–227. Springer-Verlag.

Jackson, J. and G. Mudholkar (1979): "Control procedures for residuals associated with principal component analysis." *Technometrics*, **21**, pp. 341–349.

Jalote, P. (1994): *Fault tolerance in distributed systems*. Prentice Hall, USA.

Jensen, K. and G. Rozenberg (1991): *High-level Petri Nets*. Springer Verlag.

Johanneson, G. (1994): *Object-Oriented Process Automation with SattLine*. Chartwell-Bratt Ltd.

Johnson, R. A. and D. W. Wichern (1998): *Applied Multivariate Statistical Analysis*. Prentice Hall, Upper Saddle River, New Jersey.

Johnsson, C. (1999): *A Graphical Language for Batch Control*. PhD thesis ISRN LUTFD2/TFRT--1051--SE, Department of Automatic Control, Lund Institute of Technology, Sweden.

Johnsson, C. and K.-E. Årzén (1998a): "Grafchart and batch recipe structures." In *WBF'98 — World Batch Forum*. Baltimore, MD, USA.

Johnsson, C. and K.-E. Årzén (1998b): "Grafchart and recipe-based batch control." *Computers and Chemical Engineering*, **22:12**, pp. 1811–1828.

Julier, S. J., J. K. Uhlmann, and H. F. Durrant-Whyte (1995): "A new approach for filtering nonlinear systems." In *Proceedings of the 1995 American Control Conference*, pp. 1628–1632.

Kassidas, A., J. F. MacGregor, and P. A. Taylor (1998): "Synchronization of batch trajectories using dynamic time warping." *AIChE Journal*, **44**, pp. 864–873.

Khalil, H. K. (2002): *Nonlinear Systems*. Prentice Hall, Upper Saddle River, New Jersey.

Kourti, T. (2003a): "Abnormal situation detection, three-way data and projection methods; robust data archiving and modeling for industrial applications." *Annual Reviews in Control*, **27**, pp. 131–139.

Kourti, T. (2003b): "Multivariate dynamic data modeling for analysis and statistical process control of batch processes, start-ups, and grade transitions." *Journal of Chemometrics*, **17**, pp. 93–109.

Kourti, T. and J. F. MacGregor (1995): "Process analysis, monitoring and diagnosis, using multivariate projection methods." *Chemometrics and intelligent laboratory systems*, **28**, pp. 3–21.

Kourti, T., P. Nomikos, and J. F. MacGregor (1995): "Analysis, monitoring and fault diagnosis of batch processes using multiblock and multiway pls." *Journal of Process Control*, **5**, pp. 277–284.

Kramer, M. A. (1987): "Malfunction Diagnosis Using Quantitative Models with Non-Boolean Reasoning in Expert Systems." *AIChE Journal*, **33**, pp. 130–140.

Kroonenberg, P. M. (1983): *Three-mode principal component analysis: Theory and applications*. DSWO Press, Leiden.

Kruger, U., Y. Zhou, and G. W. Irwin (2004): "Improved principal component monitoring of large-scale processes." *Journal of Process Control*, **14**, pp. 879–888.

Ku, W., R. H. Storer, and C. Georgkakis (1994): "Uses of state estimation for statistical process control." *Computers & Chemical Engineering: ESCAPE 3*, **18**, pp. S571–S575.

Ku, W., R. H. Storer, and C. Georgkakis (1995): "Disturbance detection and isolation by dynamic principal component analysis." *Chemometrics and intelligent laboratory systems*, **30**, pp. 179–196.

Lane, S., E. B. Martin, R. Kooijmans, and A. J. Morris (2001): "Performance monitoring of a multi-product semi-batch process." *Journal of Process Control*, **11**, pp. 1–11.

Larsson, J. E. (1992): *Knowledge-Based Methods for Control Systems*. PhD thesis ISRN LUTFD2/TFRT--1040--SE, Department of Automatic Control, Lund Institute of Technology, Sweden.

Larsson, J. E., J. Ahnlund, T. Bergquist, F. Dahlstrand, B. Öhman, and L. Spaanenburg (2004): "Improving expressional power and validation for multilevel flow models." *Journal of Intelligent and Fuzzy Systems*, **15:1**, pp. 61–73.

Lee, J. H. and A. W. Dorsey (2004): "Monitoring of batch processes through state-space models." *AIChE Journal*, **50**, pp. 1198–1210.

Lee, J.-M., C. K. Yoo, and I.-B. Lee (2004): "Fault detection of batch processes using multiway kernel principal component analysis." *Computers & Chemical Engineering*, **28**, pp. 1837–1847.

Lennox, B., H. Hiden, G. Montague, G. Kornfeld, and P. Goulding (2001a): "Process monitoring of an industrial fed-batch fermentation." *Biotechnology and Bioengineering*, **74:2**, pp. 125–135.

Lennox, B., K. Kipling, J. Glassey, G. Montague, M. Willis, and H. Hiden (2002): "Automated production support for the bioprocess industry." *Biotechnology Progress*, **18:2**, pp. 269–275.

Lennox, B., G. Montague, H. Hiden, and G. Kornfeld (2001b): "Moving window MSPC and its application to batch processes." In *Proc. of Computer Applications in Biotechnology*.

Li, W. and S. J. Qin (2001): "Consistent dynamic PCA based on errors-in-variables subspace identification." *Journal of Process Control*, **11**, pp. 661–678.

Li, W., H. H. Yue, S. Valle-Cervantes, and S. J. Qin (2000): "Recursive PCA for adaptive process monitoring." *Journal of Process Control*, **10**, pp. 471–486.

Lind, M. (1990): "Representing Goals and Functions of Complex Systems – An Introduction to Multilevel Flow Modeling." Technical Report. Institute of Automatic Control Systems, Technical University of Denmark.

Ljung, L. (1999): *System Identification: Theory for the User*. Prentice Hall, Upper Saddle River, New Jersey.

Lopes, J. and J. Menezes (1998): "Faster development of fermentation processes. early stage process diagnosis." In *Proc. of Foundation of Computer Aided Process Operations – FOCAPO 98, AIChE Symposium Series*, pp. 391–396.

Louwerse, D. J. and A. K. Smilde (2000): "Multivariate statistical process control of batch processes based on three-way models." *Chemical Engineering Science*, **55**, pp. 1225–1235.

Luyben, W. L. (1973): *Process Modeling, Simulation, and Control for Chemical Engineers*. McGraw-Hill.

MacGregor, J. F. and T. Kourti (1995): "Statistical Process Control of Multivariate Processes." *Control Eng. Practice*, **3:3**, pp. 403–414.

McPherson, L., E. Martin, and J. Morris (2001): "Super Model-Based Process Performance Monitoring." In *Advances in Process Control 6, IChemE*, pp. 23–30.

McPherson, L., E. Martin, and J. Morris (2002): "Super Model-Based Techniques for Batch Process Performance Monitoring." In *12th European Symposium on Computer Aided Process Engineering (ESCAPE12)*, pp. 523–528.

Meng, X., A. J. Morris, and E. B. Martin (2003): "On-line monitoring of batch processes using a PARAFAC representation." *Journal of Chemometrics*, **17**, pp. 65–81.

Miller, P., R. E. Swanson, and C. E. Heckler (1998): "Contribution Plots: A Missing Link in Multivariate Quality Control." *Applied Mathematics and Computer Science*, **8**, pp. 775–792.

Minsky, M. L. (1965): "Matter, Minds, and Models." In *Proc. of International Federation of Information Processing Congress*, pp. 45–49.

Musulin, E., M. J. Arbiza, A. Bonfill, L. Puigjaner, R. Olsson, and K.-E. Årzén (2005): "Closing the Information Loop in Recipe-Based Batch Production." In *Proceedings of European Symposium on Computer Aided Process Engineering – ESCAPE 15*.

NAMUR (1992): *NAMUR-Emphehlung: Anforderungen an Systeme zur Rezeptfaheweise (Requirements for Batch Control Systems). NAMUR AK 2.3 Funktionen der Betriebs- und Produktionsleitebene*.

Negiz, A. and A. Cinar (1997): "Statistical Monitoring of Multivariate Dynamic Processes with State-Space Models." *AIChE Journal*, **43:8**, pp. 2002–2020.

Nelson, P. R., P. A. Taylor, and J. F. MacGregor (1996): "Missing data methods in PCA and PLS: Score calculations with incomplete observations." *Chemometrics and intelligent laboratory systems*, **35**, pp. 45–65.

Nett, C. N., C. A. Jacobson, and A. T. Miller (1988): "An integrated approach to controls and diagnostics: The 4-parameter controller." In *Proceedings of the American Control Conference*, pp. 824–835.

Niemann, H. and J. Stoustrup (1997): "Integration of control and fault detection: Nominal and robust design." In *Proceedings of IFAC Safeprocess '97*, pp. 341–346.

Nilsson, A., K.-E. Årzén, and T. F. Petti (1992): "Model-based diagnosis—State transition events and constraint equations." In *Preprints IFAC Symposium on AI in Real-Time Control*. Delft, The Netherlands.

Nomikos, P. (1996): "Detection and diagnosis of abnormal batch operations based on multi-way principal component analysis." *ISA Transactions*, **35**, pp. 259–266.

Nomikos, P. and J. F. MacGregor (1994): "Monitoring batch processes using multiway principal component analysis." *AIChE Journal*, **40:8**, pp. 1361–1375.

Nomikos, P. and J. F. MacGregor (1995a): "Multi-way partial least squares in monitoring batch processes." *Chemometrics and intelligent laboratory systems*, **30**, pp. 97–108.

Nomikos, P. and J. F. MacGregor (1995b): "Multivariate SPC Charts for Monitoring Batch Processes." *Technometrics*, **37:1**, pp. 41–59.

Oakland, J. S. (1999): *Statistical Process Control*. Butterworth-Heinemann.

Olsson, R. (2002): "Exception Handling in Recipe-Based Batch Control." Licentiate thesis ISRN LUTFD2/TFRT–3230–SE. Department of Automatic Control, Lund Institute of Technology, Sweden.

Olsson, R. and K.-E. Årzén (2004): "A Modular Batch Laboratory Process." In *Proc. of 7th International Symposium on Advanced Control of Chemical Processes – ADCHEM.*

Olsson, R. and K.-E. Årzén (2000): "Exception Handling in Recipe-Based Batch Control." In *Proc. of ADPM2000 The 4th International Conference on Automation of Mixed Processes.* Dortmund, Germany.

Olsson, R. and K.-E. Årzén (2002): "Exception Handling in S88 using Grafchart." In *Proc. of World Batch Forum North American Conference 2002.* Woodcliff Lake, NJ, USA.

Olsson, R., H. Sandberg, and K.-E. Årzén (2002): "Development of a Batch Reactor Laboratory Process." In *Reglermötet 2002.* Linköping, Sweden.

Parshall, J. and L. Lamb (2000): *Applying S88 – Batch Control from a User's Perspective*. ISA – Instrument Society of America, Research Triangle Park, NC, USA.

Patton, R. J., P. M. Frank, and R. N. Clark (1989): *Fault Diagnosis in Dynamic Systems, Theory and Applications*. Prentice-Hall, Englewood Cliffs, New Jersey.

Pearson, K. (1901): "On lines and planes of closest fit to systems of points in space." *Phil. Mag.*, **2:11**, pp. 559–572.

Petti, T. F. (1992): *Using Mathematical Models in Knowledge-Based Control Systems*. PhD thesis, University of Delaware.

Piovoso, M. J. and K. A. Hoo (*Eds.*) (2002): "Control Systems Magazine – Special Issue on Chemometrics for Process Control." **22:5**.

Puig, V., J. Saludes, and J. Quevedo (1997): "Applications in fault detection and diagnosis of a new algorithm for adaptive threshold generation." In *TEMPUS Workshop on Systems Modelling, Fault Diagnosis and Fuzzy Logic Control*. Budapest, Hungary.

Qin, S. J. (2003): "Statistical process monitoring: basics and beyond." *Journal of Chemometrics*, **17**, pp. 480–502.

Ramaker, H. J., E. N. M. van Sprang, S. P. Gurden, J. A. Westerhuis, and A. K. Smilde (2002): "Improved monitoring of batch processes by incorporating external information." *Journal of Process Control*, **12**, pp. 569–576.

Rao, C. R. and S. K. Mitra (1971): *Generalized Inverse of Matrices and Its Applications*. Wiley.

Rännar, S., J. F. MacGregor, and S. Wold (1998): "Adaptive batch monitoring using hierarchical PCA." *Chemometrics and intelligent laboratory systems*, **41**, pp. 73–81.

Rosenhof, H. P. and A. Ghosh (1987): *Batch Process Automation, Theory and Practice*. Van Nostrand Reinhold.

Rotem, Y. and D. R. Lewin (2000): "Assessing the Impact of Parametric Uncertainty on the Performance of Model-Based PCA." In *Proc. of ADCHEM 2000, International Symposium on Advanced Control of Chemical Processes*.

Rotem, Y., A. Wachs, and D. R.Lewin (2000): "Ethylene Compressor Monitoring Using Model-Based PCA." *AIChE Journal*, **46:9**, pp. 1825–1836.

Ruiz, D., J. Cantón, J. Nougués, A. Espuña, and L. Puigjaner (2001): "On-line fault diagnosis system support for reactive scheduling in multipurpose batch chemical plants." *Computers and Chemical Engineering*, **25**, May, pp. 829–837.

Russell, S. A., D. G. Robertson, J. H. Lee, and B. A. Ogunnaike (2000): "Model-based quality monitoring of batch and semi-batch processes." *Journal of Process Control*, **10:4**, pp. 317–332.

Schölkopf, B., A. Smola, and K.-R. Müller (1998): "Nonlinear component analysis as a kernel eigenvalue problem." *Neural Computation*, **10**, pp. 1299–1319.

Smilde, A. K. (2001): "Comments on three-way analyses used for batch process data." *Journal of Chemometrics*, **15**, pp. 19–27.

Stamatis, D. H. (2003): *Failure Mode and Effect Analysis: FMEA from Theory to Execution*, 2nd edition. ASQ Quality Press.

Staroswiecki, M. and G. Comtet-Varga (2001): "Analytical redundany relations for fault detection and isolation in algebraic dynamic systems." *Automatica*, **37**, pp. 687–699.

Sun Microsystems (2004): "JavaCC – The Java Parser Generator." http://javacc.dev.java.net/.

Sun Microsystems, Inc (2002): "Java API for XML Processing (JAXP)." JAXP home page, http://java.sun.com/xml/downloads/jaxp.html.

Sun Microsystems, Inc (2005): "The Java Tutorial." Home page, http://java.sun.com/docs/books/tutorial/index.html.

Tikhonov, A. N. (1963): "Solution of incorrectly formulated problems and the regularization method." *Soviet Math. Dokl.*, **4**, pp. 1035–1038. English transl. of *Doklady Akademii Nauk SSSR*, **151**, 501–504.

Tittus, M. and K. Åkesson (1999): "Deadlock avoidance in batch processes." In *Proc. of IFAC World Congress*. Beijing, China.

Tucker, L. (1966): "Some mathematical notes on three-mode factor analysis." *Psychometrika*, **31**, pp. 279–311.

Tyler, M. and M. Morari (1994): "Optimal and robust design of integrated control and diagnostic modules." In *Proccedings of American Control Conference*, pp. 2060–2064.

van Beurden, I. and R. Amkreutz (2002): "Emergency Batch Landing." *InTech*, August, pp. 30–32.

Van Overschee, P. and B. De Moor (1994): "N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems." *Automatica*, **30:1**, pp. 75–93.

Vedam, H. and V. Venkatasubramanian (1999): "PCA-SDG Based Process Monitoring and Fault Diagnosis." *Control Engineering Practice*, **7:7**, pp. 903–917.

Venkatasubramanian, V., R. Rengaswamy, and S. N. Kavuri (2003a): "A review of process fault detection and diagnosis Part II: Qualitative models and search strategies." *Computers & Chemical Engineering*, **27**, pp. 313–326.

Venkatasubramanian, V., R. Rengaswamy, K. Yin, and S. N. Kavuri (2003b): "A review of process fault detection and diagnosis Part I: Quantitative model-based methods." *Computers & Chemical Engineering*, **27**, pp. 293–311.

Venkatasubramanian, V., R. Rengaswamy, K. Yin, and S. N. Kavuri (2003c): "A review of process fault detection and diagnosis Part III: Process history based methods." *Computers & Chemical Engineering*, **27**, pp. 327–346.

Wachs, A. and D. R. Lewin (1998): "Process Monitoring Using Model-Based PCA." In *Proc. 5th IFAC Symp. Dynamics and Control of Processing Systems (DYCOPS'5)*, pp. 86–91.

Wachs, A. and D. R. Lewin (1999): "Improved PCA Methods for Process Disturbance and Failure Detection." *AIChE Journal*, **45:8**, pp. 1688–1700.

Westerhuis, J. A., S. P. Gurden, and A. K. Smilde (2000): "Generalized contribution plots in multivariate statistical process monitoring." *Chemometrics and Intelligent Laboratory Systems*, **51**, pp. 95–114.

Westerhuis, J. A., T. Kourti, and J. F. MacGregor (1998): "Analysis of multiblock and hierarchical PCA and PLS models." *Journal of Chemometrics*, **12**, pp. 301–321.

Westerhuis, J. A., T. Kourti, and J. F. MacGregor (1999): "Comparing alternative approaches for multivariate statistical analysis of batch process data." *Journal of Chemometrics*, **13**, pp. 397–413.

Wise, B. and N. Ricker (1989): "Feedback strategies in multiple sensor systems." *AIChE Symposium Series*, **85:267**, pp. 19–23.

Wise, B. M., N. B. Gallagher, S. W. Butler, D. D. W. Jr, and G. G. Barna (1999): "A comparison of principal component analysis, multiway principal component analysis, trilinear decomposition and parallel factor analysis for fault detection in a semiconductor etch process." *Journal of Chemometrics*, **13**, pp. 379–396.

Wittenmark, B., K.-J. Åström, and S. B. Jørgensen (2000): *Process Control*. Department of Automatic Control, Lund Institute of Technology.

Wold, H. (1966): "Estimation of principal components and related models by iterative least squares." In *Multivariate Analysis (Ed. P.R. Krishnaiah)*, pp. 391–420. Academic Press, NY.

Wold, H. (1975): "Path models with latent variables: The NIPALS approach." In *Quantitative Sociology: International perspectives on*

mathematical and statistical model building (Ed. H.M. Blalock et al.), pp. 307–357. Academic Press, NY.

Wold, S., P. Geladi, K. Esbensen, and J. Öhman (1987): "Multi-Way Principal Components and PLS-Analysis." *Journal of Chemometrics*, **1**, pp. 41–56.

Wold, S., N. Kettaneh, H. Fridén, and A. Holmberg (1998): "Modelling and diagnostics of batch processes and analogous kinetic experiments." *Chemometrics and intelligent laboratory systems*, **44**, pp. 331–340.

Wold, S., M. Sjöström, and L. Eriksson (2001): "PLS-regression: A basic tool of chemometrics." *Chemometrics and intelligent laboratory systems*, **58**, pp. 109–130.

World Batch Forum (2003): "Batch Markup Language – BatchML." WBF home page, http://www.wbf.org/, Practices and GuideLines.

xmlBlaster.org (1999): "xmlBlaster – Message Oriented Middleware." xmlBlaster home page, http://www.xmlblaster.org/.

Yoo, C. K., J.-M. Lee, P. A. Vanrolleghem, and I.-B. Lee (2004): "On-line monitoring of batch processes using multiway independent component analysis." *Chemometrics and intelligent laboratory systems*, **71**, pp. 151–163.

Yoon, S. and J. F. MacGregor (2000): "Statistical and Causal Model-Based Approaches to Fault Detection and Isolation." *AIChE Journal*, **46:9**, pp. 1813–1824.

Yoon, S. and J. F. MacGregor (2001a): "Fault diagnosis with multivariate statistical models part I: Using steady state fault signatures." *Journal of Process Control*, **11**, pp. 387–400.

Yoon, S. and J. F. MacGregor (2001b): "Incorporation of external information into multivariate PCA/PLS methods." In *Proc. of 4th IFAC Workshop on On-Line Fault Detection and Supervision in the Chemical Process Industries (CHEMFAS-4)*. Korea.

Yue, H. H. and S. J. Qin (2001): "Reconstruction-Based Fault Identification Using a Combined Index." *Industrial & Engineering Chemistry Research*, **40:20**, pp. 4403–4414.

Zhou, K. (1998): *Essentials of Robust Control*. Prentice Hall, New Jersey.

LUND INSTITUTE OF TECHNOLOGY

Lund University

Department of Automatic Control

ISSN 0280–5316

ISRN LUTFD2/TFRT––1073––SE