DESIGN AND APPLICATION OF **PRIMAL**: A PACKAGE FOR EXPERIMENTAL MODELLING OF INDUSTRIAL PROCESSES

R.J.P. VAN DER LINDEN

DESIGN AND APPLICATION OF **PRIMAL**: A PACKAGE FOR EXPERIMENTAL MODELLING OF INDUSTRIAL PROCESSES

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de Rector Magnificus, prof. ir. M. Tels, voor een commissie aangewezen door het College van Dekanen in het openbaar te verdedigen op vrijdag 15 juni 1990 te 16.00 uur

door

RUDOLF JACOBUS PHILOMENA VAN DER LINDEN

geboren te Geldrop

Druk: Boek en Offsetdrukkerij Letru, Helmond, 04920-37797

Dit proefschrift is goedgekeurd door
de promotoren
prof. ir. O. Rademaker
en
prof. dr. ir. P. Eykhoff

Aan Francess

VOORWOORD

Het werk beschreven in dit proefschrift is uitgevoerd aan de Technische Universiteit Eindhoven in de vakgroep Systeem- en Regeltechniek van de Faculteit der Technische Natuurkunde in de periode 1984-1989.

De basis van het huidige PRIMAL pakket is gelegd in 1984 en sedertdien verder ontwikkeld door Walter Renes en Ruud van der Linden. Belangrijke bijdragen aan het vragenpagina-systeem zijn geleverd door Pim Bollen. Bij de ontwikkeling van het PRIMAL pakket, de implementatie en evaluatie van de diverse methoden en bij de toepassingen in de procesindustrie is veel werk verzet door de studenten die ik de afgelopen jaren heb begeleid: de afstudeerders Peter Berben, Rob Driessen, Paul Janssen, Peter Gerlings, Onno van het Groenewoud, Martin Hogendoorn, Tiny van Lanen, Pim van Meurs, Walter Renes en Wim Tolboom, en de stagairs Walrick Dirkx, Gert-Jan van Dijk, Rob Faessen, Ludy Hardy, Ad van Kessel, Hans Meerman, Martin Schoenaker, Frits Staals en Mark Vester.

De koppeling van PRIMAL aan het glas-feeder proces is uitgevoerd in samenwerking met de PICOS groep van Philips Glas, met name met Ton Backx en Anton Koenraads. De analyses van de data zijn onafhankelijk van en parallel aan deze groep uitgevoerd.

De metingen aan het kristallisatieproces en de destillatietoren zijn in samenwerking met Leo Steijvers en Frits van de Mortel van EXXON Chemical Holland uitgevoerd.

De commercialisatie van het PRIMAL pakket wordt ondersteund door het Instituut ITP/TUE-TNO.

Eindhoven, April 1990
Ruud van der Linden

SUMMARY

In the control of continuous production processes, the dynamical behaviour of the process often plays such an important role, that detailed knowledge of the process behaviour, preferably in the form of a sufficiently reliable model, is required. This knowledge may be obtained by theoretical and experimental means.
In experimental modelling, mathematical process models are derived from measurements of process input and output signals, which subsequently may be used in, for instance, diagnosis, process monitoring, prediction and automatic control.

This thesis describes a comprehensive strategy for experimental modelling, the development of a software package to support this strategy, and the application and evaluation of this strategy (and its methods) in industrial practice.

The developed strategy is characterised by an *interactive learning* scheme and *real-time analyses*. The scheme comprises the definition of project goals, investigation of available process knowledge, installation of equipment, experiment control, data acquisition, data conditioning, signal analysis, identification and model validation, as well as the design, test and evaluation of control systems.

In contrast to conventional "off-line" approaches to identification, the importance of real-time analyses stands central in this strategy, which therefore includes the planning and execution of the experiments as essential steps.
The real-time approach lets the user inspect and analyse the measured data immediately, accumulating knowledge about the process behaviour, and using it to improve the experiments and the analyses accordingly.
Also, the constructed models may be used as "predictors", running in parallel to and synchronised with the process, comparing the predicted with the actually observed process behaviour.
Lastly, the designed control systems may be applied in simulations, that may (gradually) shift towards actual applications, while constantly monitoring their performance.
Thus, in all project phases, the user learns effectively and may profit immediately from the acquired knowledge.

To support the experimental modelling strategy in practice, the package **PRIMAL** (Package for Real-time Interactive Modelling, Analyses and Learning) has been designed and developed. This package offers the facilities to couple to industrial processes, to carry out experiments, to analyse the data, to estimate models, and to design and test control systems, all in real-time.

For each step in the experimental modelling scheme, methods have been developed and implemented in PRIMAL. Special attention has been given to the identification step, where a variety of methods has been made available, including prediction error methods, instrumental variable methods, and new variants of output error methods described here. In the practical applications, the performance of these identification methods and of the PRIMAL package as a whole have been evaluated.

PRIMAL has been applied to several industrial processes, of which the obtained results on a glass production process, a para-xylene crystallization process and a toluene-xylene distillation process are described in this thesis.
The results show that the developed experimental modelling strategy, and its modern multivariable methods, may be applied successfully to industrial processes and may lead to valuable physical insight and improved control.

CONTENTS

CHAPTER 1 INTRODUCTION

1.1 Trends in process control

In the course of the past decades, industrial processes have been automated at a high pace. Limiting the scope to continuous production processes as mainly found in the process industries, we see that computer-based automatic control systems are now commonplace, [Åström, 1985; Van Cauwenberghe, 1985].

In parallel to the development of automatic control systems, much research has been done on modelling and control, concentrating mainly on (multivariable) linear systems, [Eykhoff, 1974, 1981; Åström & Wittenmark, 1984; Ljung, 1987, 1988; ..].

Despite the abundance of modelling and control design techniques, the classical PID-controller is still widely used in single loops or cascades in the process industry, and the controllers are often still tuned using only a few characteristics of the process.
Recently, however, there is a new trend towards application of Multi-Input Multi-Output (MIMO) process modelling and control design techniques in industry.

A first reason for this trend is that sharper demands are made on the production process in terms of quality and reproducibility, energy and raw materials consumption, throughput, flexibility, and pollution, which can, hopefully, be achieved by more sophisticated control, based on mathematical models of the process.

Secondly, the theory of modelling and control system design of linear systems has matured. In identification theory several attempts to unify the "bag of tricks" [Eykhoff, 1974; Ljung, 1987] have shown that numerous different identification techniques may be regarded as special cases of a few general techniques.
Based upon such techniques commercial tools become available for industrial practice.

Thirdly, modern instrumentation systems provide sufficient programmability, computing power and data storage to meet the technical requirements for the implementation of advanced control.
Furthermore, the possibility to automatically collect and analyse (large amounts of) process data is of fundamental importance to techniques for experimental analysis.

This thesis concentrates on methods, techniques and tools that may be used in modelling and control of continuous and batch-wise continuous production processes. It introduces a strategy and describes an environment and tools for carrying out real-time experimental analyses of the (dynamical) process behaviour.

1.2 Some notes on modelling

If the dynamic behaviour of a process is not well understood, for instance due to complex interactions between several process variables or as a result of noise or external disturbances, it may be difficult to predict and control future behaviour. In these cases, developing a mathematical model reflecting the process properties of interest, may contribute to understanding and controlling process behaviour.

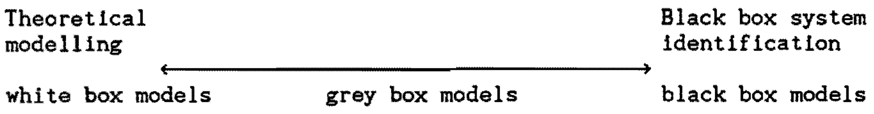
In the process design phase, models of its dynamics may be used to ensure a requested dynamical behaviour.
For existing processes, models may be used for diagnosis, for monitoring the process to detect deviations between the actual and expected process behaviour in an early stage, and for prediction and control of process behaviour in the case of set point changes and disturbances. The performance of a process may critically depend on the ability of the control system to reduce the effect of disturbances and to keep the operation conditions within the desired range.

A way to order the various approaches to mathematical modelling, is by picturing them on a scale with at the ends the two extremes: at one side "theoretical modelling" and on the other side "black box system identification".

In theoretical modelling, basic laws from physics (conservation laws), thermodynamics, chemistry, together with empirical relations are used to construct a model.

In black box system identification, a model class is posited, which usually does not reflect the internal structure of the process. The model parameters are estimated from the observed data, using some optimisation technique for minimising a criterion of the misfit between the behaviour of the model and the behaviour of the real process.
Since the models do not reflect the internal structure of the process they are often called "black box" models. Only their input-output behaviour is considered important.

Analogous to black box models, it has become common to define "white box" models as the models resulting from theoretical modelling.



In practice, the two extremes are never used, but a sensible combination of the two approaches is chosen, resulting in more or less "grey" models.

Some characteristics of these approaches:

Theoretical modelling:

- Constructing a theoretical model of an industrial process requires detailed knowledge about the physical and chemical phenomena involved. As a result, model construction requires expert knowledge that may be unavailable and is usually very time-consuming.
- In several stages during the construction of the model, simplifications or assumptions are necessary in order to be able to write down mathematical equations. Their influence on the accuracy of the model is often difficult to predict and no direct measures of the loss of accuracy are available.
  As a result, experiments are still needed to verify the model. If the verification reveals model errors, model-adjustment is needed, for instance by adaptation of parameter values.
- The model not only describes the input-output properties of the process, but also its internal behaviour. The states and parameters in the model customarily have a direct physical interpretation.
- Usually, the model has a wide validity region, depending on the assumptions made when deriving the equations. The model may be used in the process design phase and for different process sizes. On the other hand, the modelling is very process specific: another type of process may require a basically different model.
- Since the resulting model is often a complex set of (P)DV's and algebraical equations, its use in simulations may be numerically difficult and time-consuming, and the model may be inappropriate for use in real-time.
- If, as hinted at above, the model contains unknown or ill-known parameters, experiments are still needed to fit the model to the experimental data. This usually results in complicated constrained non-linear optimisation problems.

Black box system identification:

- Black box models only require knowledge about the input-output behaviour of the process, but no detailed knowledge about its internal behaviour.
- Since the model is not based on physical principles, it is generally accepted that it yields little physical insight into the process behaviour.
- For constructing the model it is necessary to carry out experiments. The process must therefore exist, be well instrumented, and the relevant input and output variables must be measurable and/or excitable.
- The quality of the black box model depends critically on the information contents of the process data used for its construction. Therefore, data acquisition and experiment design are key factors for a successful approach.

- Usually, the resulting model has a limited validity range. It is accurate only for a specific process, in a certain operating range. Within this range, however, the model may be very accurate.
  Since the model is estimated from experimental data, it is usually feasible to derive direct estimates of its accuracy. However, there is no guarantee that the model will remain viable under different process conditions.
- The techniques used in the identification approach are not process-specific and may be applied to widely different types of arbitrarily complex processes.
- Efficient numerical techniques are available for estimating the parameters of linear black box models from the observed data.
- Black box models are usually sufficient for prediction and control. Because of their simple structure they lend themselves better to real-time application.
- The identification approach cannot be used in the process design phase.

This thesis describes a methodology and techniques for experimental modelling.
The process of finding a suitable model by experimental modelling comprises the following steps:

- Design of experiments.
- Data acquisition: the collection of data from the process.
- Selection of a set of candidate models (the model set).
- A choice of identification techniques for selecting the best model from the model set.
- Model validation: assessment of the quality of the estimated model.

The actual methods that are considered are restricted to black box techniques. The main motivation for this approach is a pragmatic one. The most important advantage of black box techniques are the applicability to widely different processes and the ability to generate a model within a short timespan. If the purpose of a model is prediction or control, black box models usually prove to be effective.
Since physical knowledge about the process is extensively used in experiment design and model structure selection, we prefer to call this approach "dark-grey" modelling.

In experimental modelling, a point of great concern is the dependence on "good" data, that are representative of the process behaviour. Usually, this means that experiments must be carried out which involve superimposing test signals on the process inputs.

We assume that data, collected from the process, consists of a set of signals, sampled equidistantly in the time.
The structure of the models under consideration is restricted to discrete-time, linear, causal, time-independent, finite order, multi-input, multi-output models, as described in Chapter 2.
The restriction to this class limits the type of processes to which

the techniques can be applied. Linear models, however, may be used successfully for description of the process near an operating point. The advantage of this approach is the availability of a rigorous mathematical theory and a great variety of efficient numerical procedures.
Some types of processes may, however, present insuperable problems, for instance, processes subject to hysteresis, pronounced (dynamical) non-linearity or abrupt changes in the dynamical behaviour.

1.3 Scope of this thesis

In this thesis we will:

- Discuss a methodology for experimental analysis of production processes in the process industry.
- Describe a Package for Real-time Interactive Modelling, Analyses and Learning (**PRIMAL**) to implement the methodology.
- Discuss the application of the methodology and of PRIMAL in the experimental analysis of several industrial processes.
- Assess the usefulness of established identification techniques and discuss effective variants.

The thesis is structured as follows:

Chapter 2 presents the theoretical framework and those concepts that are needed in Chapters 3 and 5. Identification is considered in terms of approximate modelling.
Chapter 3 discusses the experimental modelling strategy.
Chapter 4 considers the implementation of the strategy in PRIMAL, which provides a platform for experimental modelling and control design methods and which has been specifically designed for application in industry.
Chapter 5 discusses the identification methods made available in PRIMAL.
Chapter 6 discusses the application of the experimental modelling strategy to three cases in the process industry.
In Chapter 7 the final conclusions are summarised.

In summary, the principal contributions described in this thesis are:

- The development of a comprehensive scheme for experimental modelling, that is based on interactive learning strategy for process analyses in real-time.
  The scheme covers preliminary investigations, connecting to the process, experiment design, data acquisition, data conditioning, signal analyses, identification, model validation, control system design and testing.
- The PRIMAL package, which implements the interactive (real-time) strategy.
- The application and testing of the experimental modelling strategy in several cases in industrial practice.

CHAPTER 2 THEORETICAL FRAMEWORK

In Chapter 1, identification has been introduced as a way to obtain a mathematical model of a process on the basis of experimental data. Since usually the physical relations governing the process behaviour are more complex than the proposed model and as the data obtained from the process may describe only part of its behaviour, identification should be considered as a way to generate an *approximate* model.

This chapter summarises selected concepts and definitions to serve as a background for the discussion of the experimental modelling scheme in Chapter 3 and the identification methods in Chapter 5, following the lines of "classical" identification theory, as formalised by Ljung [1976, 1977, 1983, 1987] and Söderström & Stoica [1983a; Söderström, 1989]. The experienced reader may proceed directly with Chapter 3.

Section 2.1 discusses the approximate modelling problem. Section 2.2 presents the different model representations used to describe linear systems in PRIMAL. In Sections 2.3 and 2.4 the basic system description and the resulting predictor models are introduced. Section 2.5 concentrates on the approximate modelling aspects of the predictor models. In Section 2.6 different parametrisations are introduced that are used in the PRIMAL identification methods discussed in Chapter 5.

2.1 Approximate modelling

In order to stress the notion that identification is considered as approximate modelling, it is useful to distinguish between the model that is assumed to have generated the data, and the model derived (estimated) from the observed data.
The term Data Generating Model (DGM) may be introduced, cf. [Janssen, 1988] to describe how the outputs y(t) are constructed from the inputs u(t) and disturbances e(t). The DGM is a theoretical construct with the main purpose to serve as a basis for discussing identification methods and for analysing their properties.

Identification may be viewed as finding a model that describes sufficiently well the dynamical relation between the observed input and output data sequences. Thus, it requires a rule for generating a model residual from the experimental data and for minimising some scalar measure of the residuals (i.e. a Residual Generating Model (RGM), cf. [Janssen, 1988]). We will use the concept of a "predictor model" as formalised by Ljung [1987], i.e. a rule of predicting the output of a process on the basis of past data of the measured inputs and outputs.

2.2 Model representations

In this section we introduce different ways to represent models of linear, discrete time, time-invariant, finite dimensional, multivariable systems, briefly discussing their relations.
This subject is treated extensively in the literature [Kailath, 1980], and consequently, we restrict ourselves to the definitions and terminology needed later.

We will discuss causal systems with p inputs, $u(t) \in \mathbb{R}^p$, and q outputs, $y(t) \in \mathbb{R}^q$, which are functions of the discrete time t: $t \in \mathbb{Z}^+$.

Vector difference equations

A common description of a linear dynamical system is given by a backward vector difference model:

$$\begin{aligned} &A_0y(t) + A_1y(t-1) + \dots + A_{na}(t-na) = \\ &\qquad B_0u(t) + B_1u(t-1) + \dots + B_{nb}u(t-nb) \end{aligned} \qquad (2.2.1)$$

where $B_i \in \mathbb{R}^{qxp}$, $i = 0..nb$, and $A_i \in \mathbb{R}^{qxq}$, $i = 0..na$, are matrices with constant coefficients. Equation (2.2.1) expresses the output y(t) in terms of the input u(t) and previous values of inputs and outputs.

Introducing the backward shift operator q^{-1}, $q^{-1}u(t) = u(t-1)$, this model may be written as:

$$A(q^{-1})y(t) = B(q^{-1})u(t) \qquad (2.2.2)$$

$$A(q^{-1}) = \sum_{i=0}^{na} A_i q^{-i}$$

$$B(q^{-1}) = \sum_{i=0}^{nb} B_i q^{-i}$$

where $A(q^{-1})$, $B(q^{-1})$ are matrix functions in the shift operator q^{-1}.

Using the z-transform a similar form is obtained:

$$A(z^{-1})y(z) = B(z^{-1})u(z) \qquad (2.2.3)$$

where $A(z^{-1})$, $B(z^{-1})$ are polynomial matrices in z^{-1}.

Models with this structure are usually called ARMA (Auto Regressive Moving Average) models or Matrix Fraction Descriptions (MFD's), depending on their formulation in either the forward or the backward shift operator. We will not use this distinction, and only discuss models in the backward shift operator. This form makes the models suited for direct use in simulation and prediction.

The model (2.2.1) describes a causal (proper) system if A_0 is non-singular (i.e. $\det(A_0) \neq 0$). If we further require that A is *monic*, $A_0 = I$, we obtain a straightforward expression for y(t):

$$y(t) = -\sum_{i=1}^{na} A_i y(t-i) + \sum_{i=0}^{nb} B_i u(t-i) \tag{2.2.4}$$

Additionally we often require the system to be strictly proper, i.e. $B_0 = 0$, which is a natural assumption for most sampled data systems.

Transfer function models

An alternative way to represent the system is by its input-output transfer operator (transfer function) $C(q^{-1})$:

$$y(t) = C(q^{-1})u(t) \tag{2.2.5}$$

with $C(q^{-1})$ a qxp rational matrix function in q^{-1}.

Using the Laurent expansion $C(q^{-1})$ can also be written as:

$$C(q^{-1}) = \sum_{i=0}^{\infty} C_i q^{-i} \quad , C_i \in \mathbb{R}^{qxp} \tag{2.2.6}$$

The system is strictly proper if $C_0 = 0$.
This transfer function representation is related to the difference equation. We may rewrite (2.2.1) to:

$$y(t) = A^{-1}(q^{-1})B(q^{-1})u(t) := C(q^{-1})u(t) \tag{2.2.7}$$

Matrix fraction descriptions

The term MFD was already mentioned in relation to the vector difference equation. Put more formally, we define a MFD as follows, cf. [Kailaith 1980]: Let C(z) be a rational matrix of dimension qxp and A(z) a qxq non-singular polynomial matrix, B(z) a qxp polynomial matrix, then $C(z) = A^{-1}(z)B(z)$ is a left MFD of C(z).
The *degree* of the MFD is defined as the degree of the determinant of A(z): deg det A(z).

A MFD is not a unique representation of the system's transfer function. Multiplying $A(z)$ and $B(z)$ by any non-singular polynomial matrix $W(z)$ yields a MFD with exactly the same transfer function. If $W(z)$ is not unimodular it will affect the degree of the MFD. The MFD is said to be *left coprime* if $A(z)$ and $B(z)$ only have unimodular left divisors.
Still, there are infinitely many left coprime MFD's describing the same system, because transformation of $\{A(z),B(z)\}$ with any unimodular $W(z)$ will produce another MFD with the same degree.

The matrix fraction description is the extension to the multivariable case of the numerator and denominator polynomials in the transfer function in the Single Input, Single Output (SISO) case. Analogously the concept of zeros and poles may be extended to the MIMO case on the basis of the Smith-McMillan form.

Markov parameter models

A linear system may be described completely by its impulse response. For a MIMO system the value $m_{ji}(k)$ of the impulse response of output j at time instant k, as a result of an impulse $\delta(k)$ on input i may be written into the matrix:

$$M(k) := [\ m_{ji}(k)\ ], \qquad k = 0,1,.. \qquad (2.2.8)$$

The matrix $M(k)$ is called the k-th Markov parameter.
The Markov parameters are a unique description of the transfer function. However, an infinite number of parameters is required to exactly describe it. For a finite dimensional system the Markov parameters are related, [Backx, 1987].
The response $y(t)$ of the system to an arbitrary input $u(t)$ may be written as a convolution sum:

$$y(t) = \sum_{k=0}^{\infty} M(k)u(t-k) \qquad (2.2.9)$$

For a strictly proper system $M(0) = 0$.
Comparing (2.2.6) with (2.2.9) shows the immediate relation between the transfer function and the Markov parameters.

State space models

The state space representation describes an n-th order system by n first order equations:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \qquad (2.2.10)$$

with a state vector $x(t) \in \mathbb{R}^n$ and the matrices $\{A \in \mathbb{R}^{nxn}, B \in \mathbb{R}^{nxp}, C \in \mathbb{R}^{qxn}, D \in \mathbb{R}^{qxp}\}$ defining the model.
The state space equations describe a causal system. If D=0 the system is strictly proper.
The state space representation and the transfer function are related according to:

$$G(z) = C(zI - A)^{-1}B + D \qquad (2.2.11)$$

So, an MFD may be transformed to a state space representation (realisation) and vice versa, [Wolovich, 1974].
Any realisation of order n = deg det A(z) of an MFD $A^{-1}(z)B(z)$ is a minimal realisation if and only if the MFD is left coprime, [Kailath, 1980].

Relations between the models

In the discussion above it was briefly indicated how the different model representations are related. For a more formal treatment of the relations refer to [Hannan, 1988; Kailath, 1980; Janssen, 1988; and the references therein].
The PRIMAL package extensively uses the state-space and (left) MFD's for representing systems. The polynomial matrices occurring in the MFD are stored as three-dimensional matrices.

2.3 Data generating models for identification

For the purpose of identification the basic model types introduced in Section 2.2 are parametrised and extended with a term representing the part of the output that can not be explained by the input, which is usually considered to be of a stochastic nature.

Let $\mathcal{M}$ be a model set parametrised by $\theta \in \mathcal{D}m$. This may be written as: $\mathcal{M} = \{\mathcal{M}(\theta) \mid \theta \in \mathcal{D}m\}$, cf. Appendix A.

We now introduce the general data generating model:

$$\mathcal{M}(\theta): y(t) = G(q^{-1};\theta)u(t) + H(q^{-1};\theta)e(t) \qquad (2.3.1)$$

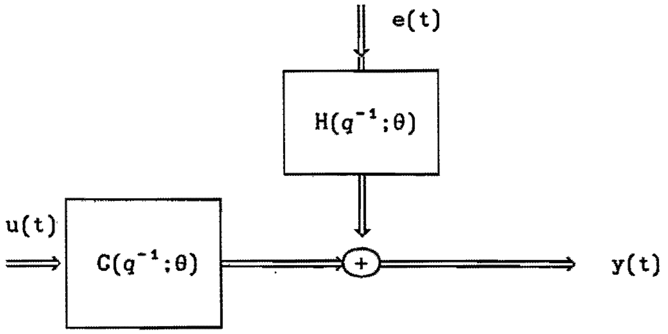
$$G(q;\theta) = G_0(\theta) + G_1(\theta)q^{-1} + G_2(\theta)q^{-2} + .. \qquad (2.3.2)$$
$$H(q;\theta) = H_0(\theta) + H_1(\theta)q^{-1} + H_2(\theta)q^{-2} + .. \qquad (2.3.3)$$

where:

t	: the discrete time, $t \in \mathbb{Z}^+$
e(t)	: a sequence of independent and identically distributed random variables with zero mean (a white noise)
θ	: the $n\theta$ dimensional parameter vector
G(.)	: a linear (qxp) dimensional filter, depending on θ
H(.)	: a linear (qxq) dimensional filter, depending on θ

Usually the set of admissible models is restricted to $\theta \in \mathfrak{D}_m$.

$$\mathfrak{D}_m = \{ \theta \mid \; C_0(\theta) = 0, \quad H_0(\theta) = I \text{ (monic)}, \quad H^{-1}(z) \text{ and } H(z) \text{ analytic for } |z| \geq 1, \quad H^{-1}(q;\theta)C(q;\theta) \text{ stable} \} \tag{2.3.4}$$



The model consists of a deterministic part $C(q^{-1};\theta)u(t)$ and a stochastic part $H(q^{-1};\theta)e(t)$. All disturbances inflicting the process, like uncontrolled inputs and measurement noise, are lumped together in this term and are thus assumed to be generated by a noise colouring filter $H(q^{-1};\theta)$ from a white noise e(t).
The model is therefore completely described by the filters G and H and the probability density function of the noise. In the sequel e(t) is usually characterised only by its first and second order properties.

$$Ee(t) = 0$$
$$Ee(t)e^T(s) = \Lambda\delta_{t,s} \tag{2.3.5}$$

The linear, finite order, time-invariant data generating model (2.3.1) is the basis for most identification methods, mainly owing to its nice mathematical properties. In practice, the model restrictions limit their application.
As stated in Chapter 1, application is restricted to continuous production processes, operating in a limited range around an operating point. In this case the process may be approximated by a linear, time-invariant model. However, the process needs not to be in the model set to achieve a satisfactory result. For approximate modelling the optimality of the techniques based on linear models is lost, but yet the approximation may be adequate.

Especially the modelling of the disturbances as filtered zero-mean white noise seems artificial. This description of noise is supported by the central limit theorem and the spectral factorisation theorem; cf. [Anderson & Moore, 1979], which states that any nonsingular rational spectral density function $\phi(\omega)$ can be written as:

$$\phi(\omega) = \frac{1}{2\pi} H(e^{-i\omega})\Lambda H^T(e^{i\omega}) \qquad (2.3.6)$$

$H(q^{-1})$ and $H^{-1}(q^{-1})$ asymptotically stable
$H(0) = I$

As a consequence the signal with spectral density $\phi(\omega)$ can be written as $y(t) = H(q^{-1})e(t)$, $Ee(t)e^T(s) = \Lambda\delta_{t,s}$

Yet, the disturbances encountered in practice will rarely fit this format. One may expect deterministic disturbance components and non-linear effects. The description of the noise should therefore be considered as a vehicle to arrive at predictor models with nice properties.

2.4 Predictor models

For linear, time-invariant systems a predictor model may be defined as:

Definition 2.4.1 [Ljung, 1987]
A *predictor model* of a linear, time-invariant system is a filter:

$$\hat{y}(t|t-1) = W_u(q)u(t) + W_y(q)y(t) \qquad (2.4.1)$$

where $W_u(q)$ and $W_y(q)$ are stable linear filters . □

Note that this definition does not specify the behaviour of the prediction error:

$$\varepsilon(t) = y(t) - \hat{y}(t|t-1) \qquad (2.4.2)$$

Definition 2.4.2
A *k-step-ahead predictor model* is a predictor model for y(t) that only depends on data Z^{t-k}, $k \geq 1$.
$Z^{t-k} := \{u(n),y(n) \mid n = 1,2,\dots,t-k\}$ □

Straightforward analysis shows how a one-step-ahead prediction error model may be derived for the general model (2.3.1).
We assume that u(t) is uncorrelated with e(s) for t<s. This is a natural assumption for open loop and causal feedback situations.

First (2.3.1) is rewritten to:

$$\begin{aligned} y(t) &= G(q;\theta)u(t) + H(q;\theta)e(t) \\ &= G(q;\theta)u(t) + [H(q;\theta)-I]e(t) + e(t) \\ &= G(q;\theta)u(t) + [H(q;\theta)-I]H^{-1}(q;\theta)[y(t)-G(q;\theta)u(t)] + e(t) \\ &= H^{-1}(q;\theta)G(q;\theta)u(t) + [I-H^{-1}(q;\theta)]y(t) + e(t) \qquad (2.4.3) \end{aligned}$$

The first two terms are known at time t, since they contain only past data. The last term e(t) is unpredictable.
An estimate for y(t) may be generated by taking the conditional expectation of y(t). Using the properties of e(t) (2.3.5) this conditional expectation may be written as:

$$\hat{y}(t|t-1;\theta) = H^{-1}(q;\theta)G(q;\theta)u(t) + [I-H^{-1}(q;\theta)]y(t) \qquad (2.4.4)$$

This predictor fits the definition of a predictor model:

$$\begin{aligned} W_u(q;\theta) &= H^{-1}(q;\theta)G(q;\theta) \\ W_y(q;\theta) &= I - H^{-1}(q;\theta) \qquad (2.4.5) \end{aligned}$$

The predictor (2.4.4) is also optimal in the mean square sense.
Taking an arbitrary linear predictor s(t) for y(t), depending only on past input and output data, and computing the prediction error covariance matrix shows:

$$\begin{aligned} &E(s(t)-y(t))(s(t)-y(t))^T = \\ &E(s(t)-\hat{y}(t|t-1;\theta))(s(t)-\hat{y}(t|t-1;\theta))^T + Ee(t)e^T(t) \geq \Lambda \qquad (2.4.6) \end{aligned}$$

Equality is achieved for $s(t) = \hat{y}(t|t-1;\theta)$

Stability of the predictor requires that the filters $H^{-1}(q;\theta)$ and $H^{-1}(q;\theta)G(q;\theta)$ are stable. The admissible set of parameters is therefore restricted to (2.3.4).

Prediction error methods

The term Prediction Error Method (PEM) is generally used for methods that generate a parameter estimate by minimising a scalar function of the prediction errors. Since the prediction errors are generated by filtering the data (2.4.4) we may write:

$$\hat{\theta}_N = \underset{\theta \in \mathcal{D}m}{\arg\min} \; V_N(\theta, Z^N) \qquad (2.4.7)$$

where V_N is some scalar function of the prediction errors.

A PEM is thus defined by:
- a choice of model structure (a predictor model)
- a criterion function V

Common criterion functions are:

$$V_N(\theta,Z^N) = \det [\ R_N(\theta,Z^N)\ ] \qquad (2.4.8)$$

and:

$$V_N(\theta,Z^N) = \mathrm{tr}\ \Omega R_N(\theta,Z^N) \qquad (2.4.9)$$

with:

Ω a positive definite weighting matrix,

$$R_N(\theta,Z^N) = \frac{1}{N}\sum_{t=1}^{N} \varepsilon(t,\theta)\varepsilon^T(t,\theta) \qquad (2.4.10)$$

Remark: Söderström [1989] shows that a PEM with criterion function (2.4.9) coincides with the Maximum Likelihood Estimation of θ in case of jointly Caussian noise e(t).

Remark: The criterion (2.4.9) asymptotically leads to the same parameter estimates and parameter covariance estimates (2.4.8) only if the weighting matrix $\Omega \equiv \Lambda^{-1}$. In iterative and recursive identification methods, Λ can be approximated by the covariance matrix of the innovations $\varepsilon(t,\theta)$.

Linear regression models and instrumental variable methods

If the prediction error is linear in the parameters a quadratic criterion function can be minimised by the ordinary least squares method. In this case the prediction error can be rewritten to:

$$\varepsilon(t) = y(t) - \varphi^T(t)\theta \qquad (2.4.11)$$

where φ is a vector containing only past values of the inputs and outputs. Minimising V with respect to θ results in the Least Squares (LS) solution:

$$\hat{\theta}_N = \mathrm{sol}\ \{ \frac{1}{N} \sum_{t=1}^{N} \varphi(t)[y(t)-\varphi^T(t)\theta] = 0 \} \qquad (2.4.12)$$

which leads to:

$$\hat{\theta}_N = [\ \frac{1}{N}\sum_{t=1}^{N} \varphi(t)\varphi(t)^T\ ]^{-1}\ [\frac{1}{N}\sum_{t=1}^{N} \varphi(t)y(t)\ ] \qquad (2.4.13)$$

This equation can be solved explicitly and yields the global minimum of the criterion function. However, it is well known that only under rather unrealistic assumptions the estimate is unbiased.

Assuming that y(t) is generated by a DGM:

$$y(t) = \varphi^T(t)\theta_o + v_o(t) \qquad (2.4.14)$$

and substituting y(t) in equation (2.4.13) leads to:

$$\hat{\theta}_N = \theta_o + [\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi(t)^T]^{-1} [\frac{1}{N}\sum_{t=1}^{N}\varphi(t)v_o(t)] \qquad (2.4.15)$$

which is biased unless the first matrix is non-singular and $\varphi(t)$ is uncorrelated with the disturbances $v_o(t)$. The first condition is generally satisfied if the input is persistently exciting of sufficient order. The second condition is satisfied only if:

$v_o(t)$ is a white noise

or,

the input $\{u(t)\}$ is independent of $\{v_o(t)\}$ and the model includes only Moving Average (MA) parameters.

The bias problem may be overcome in several ways. One approach is to model the disturbances as filtered white noise and to include the parameters of the filter $H(q^{-1})$ in the parameter vector θ. The linearity of the regression model is generally lost, so that a numerical optimisation method must be used to estimate θ.
Another approach is to find a vector of instruments $\eta(t)$ that is correlated with $\varphi(t)$, but uncorrelated with $v_o(t)$.

Methods to construct a vector of instruments $\eta(t)$ satisfying the above requirements and solving the equations:

$$\hat{\theta}_N = \underset{\theta\in\mathcal{D}m}{\mathrm{sol}} \{ \frac{1}{N}\sum_{t=1}^{N} \eta(t)[y(t)-\varphi^T(t)\theta] = 0 \} \qquad (2.4.16)$$

are called Instrumental Variable (IV) Methods.
A suitable vector of instruments can be constructed by filtering input and output data:

$$\eta(t) = W_u(q^{-1})u(t) + W_y(q^{-1})y(t) \qquad (2.4.17)$$

IV-methods and extended IV-methods implemented in PRIMAL are discussed in Chapter 5.

2.5 Asymptotic properties and approximate modelling

The behaviour of parameter estimation methods may be investigated within a stochastical framework, which implies that the properties of the estimation methods are determined for ensembles of measurements. In practice, however, we usually have just one record of data from a given process.

Under rather general assumptions, see Appendix A, the first and second order moments of a stochastic process $s(t) = m(t) + v(t)$, with $m(t)$ a bounded deterministic signal and $v(t)$ a filtered white noise, may be computed from a single realisation $\{s(t)\}$ w.p. 1 as the number of samples $N \to \infty$.
As a consequence, considering only first and second order moments, the stochastic and deterministic approaches lead to the same result. To eliminate the distinction between the two approaches the generalised expectation operator $\bar{E}$ is introduced.

$$\bar{E}s(t) := \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} Es(t) \tag{2.5.1}$$

The operator $\bar{E}$ may be applied both to stochastic and deterministic signals.

If $u(t)$ is regarded as a deterministic signal, the output $y(t)$ (2.3.1) has a deterministic and a stochastic component. As a result $\{y(t)\}$ is not a stationary stochastic process, i.e. the moments are time-dependent. We may treat $\{y(t)\}$ as a generalised weakly stationary process or as a quasi-stationary signal, see Appendix A, i.e. a single realisation of the stochastic process.

The correlation function $R_s(\tau)$ is defined as:

$$R_s(\tau) := \bar{E}s(t)s(t-\tau) \qquad \tau\in\mathbb{Z} \tag{2.5.2}$$

In the case $\{s(t)\}$ is a zero mean stationary stochastic process $R_s(\tau)$ is called the covariance function of s.
Similarly the cross-correlation function is defined as:

$$R_{sw}(\tau) := \bar{E}s(t)w(t-\tau) \qquad \tau\in\mathbb{Z} \tag{2.5.3}$$

with $\{s(t)\},\{w(t)\}$ jointly quasi-stationary.
The spectrum of $s(t)$ is defined as:

$$\Phi_s(\omega) := \sum_{\tau=-\infty}^{\infty} R_s(\tau)e^{-i\tau\omega} \tag{2.5.4}$$

and analogously the cross-spectrum $\Phi_{sw}(\omega)$ is defined.

General results on asymptotic behaviour of PEM's

Consider the one-step-ahead predictor (2.4.2) and a quadratic criterion function (2.4.9), then under rather weak assumptions [Ljung, 1987] the criterion function converges uniformly in $\theta\in\mathcal{D}_m$ as $N \longrightarrow \infty$:

$$V_N(\theta) \longrightarrow \bar{V}(\theta) := \bar{E}\varepsilon(t,\theta)^T\Omega\varepsilon(t,\theta) \tag{2.5.5}$$

Furthermore,

$$\hat{\theta}_N \longrightarrow \bar{\theta} := \arg \min_{\theta \in \mathcal{D}m} \bar{V}(\theta) \tag{2.5.6}$$

This result shows that as N approaches infinity the parameter estimates converge to minimizing arguments of the asymptotic criterion function. If the "true" system is not represented in the model set an element in the model set is found, that is closest to the true system, in the sense of a minimal prediction error variance. In this case the model found will depend on the data set, which is illustrated by the examples 2.3 and 2.4 in Söderström [1989].

Under the assumption that the data can be described by (2.3.1), i.e. the true system is in the model set, it can be shown that the PEM is consistent and the parameter estimates $\hat{\theta}_N$ are asymptotically Gaussian distributed.

$$\sqrt{N}\,(\hat{\theta}_N - \bar{\theta}) \longrightarrow AsN(0,P) \tag{2.5.7}$$

$$P = [\bar{V}''(\bar{\theta})]^{-1}[\lim_{N\to\infty} N\bar{E}\{V_N'(\bar{\theta})\}^T\{V_N'(\bar{\theta})\}][\bar{V}''(\bar{\theta})]^{-1}$$

For Gaussian distributed disturbances the optimal PEM is asymptotically efficient, i.e. the parameter covariance equals the Cramér-Rao lower bound.

General results on the asymptotic behaviour of IV-methods

Defining:

$$f_N(\theta) := \sum_{t=1}^{N} \eta(t)\epsilon(t,\theta) \tag{2.5.8}$$

(2.4.16) can be written as:

$$\hat{\theta}_N = \operatorname*{sol}_{\theta \in \mathcal{D}m} \{ f_N(\theta) = 0 \}$$

Restricting $\eta(t)$ to be an instrumental vector that is obtained by linear filtering of past data:

$$\eta(t) = W_u(q,\theta)u(t) + W_y(q,\theta)y(t) \tag{2.5.9}$$

with $\{W_u(q,\theta),\ W_y(q,\theta) \mid \theta \in \mathcal{D}m \}$, a family of uniformly stable filters, it can be shown that the parameter estimate converges uniformly as $N \to \infty$.

$$f_N(\theta) \longrightarrow \bar{f}(\theta) := \bar{E}\eta(t)\varepsilon(t,\theta) \qquad (2.5.10)$$

Analogously the set of solutions $\hat{\theta}_N$ of $f_N(\theta)=0$ converge to the set of solutions of $\bar{f}(\theta)=0$.

Under rather weak conditions it can be shown that the parameter estimates are asymptotically Gaussian distributed, [Ljung, 1987].

2.6 Model parametrisations

Before presenting different ways of parametrising systems let us introduce some terminology.

Definition 2.6.1 Model structure
A model structure P: $\mathcal{D}m \rightarrow \mathcal{M}$ is a differentiable mapping from a connected, open subset $\mathcal{D}m(\theta)$ to $\mathcal{M}$ such that the gradient of $P(\theta)$ with respect to θ is a stable transfer function. □

The differentiability is required to assure that the gradients of the predicted output with respect to θ exist and are stable, which is necessary for the functioning of prediction error methods.

Definition 2.6.2 Global identifiability with respect to θ^*.
Let $\mathcal{M}$ be a model set and $\sim$ be an equivalence relation on $\mathcal{M}$. Let P: $\mathcal{D}m \rightarrow \mathcal{M}$ be a parametrisation.
P is called globally identifiable with respect to $\sim$ at θ^* if
$\forall_{\theta \in \mathcal{D}m}\ [\ P(\theta) \sim P(\theta^*) \Rightarrow \theta = \theta^*\ ]$ □

Definition 2.6.3 Global identifiability
P is called globally identifiable with respect to $\sim$ if P is globally identifiable with respect to $\sim$ at almost all $\theta \in \mathcal{D}m$. □

The nature of $\sim$ has not yet been specified. If talking about predictor models it is convenient to define equivalence as predictor transfer function equivalence.

The definition of identifiability has the nice property, compared to parameter identifiability as defined by Söderström [1989] , that it does not depend on assumptions concerning the DGM, but solely on the parametrisation. The identifiability property is important for numerical optimisation. It ensures the existence of isolated optima. The dataset must be informative enough to distinguish between different models in the model set.

We will now briefly discuss model structures used in identification methods in PRIMAL.

ARX models

$$A(q^{-1};\theta)y(t) = B(q^{-1};\theta)u(t) + e(t) \qquad (2.6.1)$$

The corresponding predictor model is linear in the parameters (if A and B are properly parametrised):

$$y(t|t-1;\theta) = (I-A(q^{-1};\theta))y(t) + B(q^{-1};\theta)u(t)$$

The model structure is globally identifiable.
Recursive and direct methods for this model structure are discussed in Chapter 5.

FIR models

$$y(t) = B(q^{-1};\theta)u(t) + e(t) \qquad (2.6.2)$$

The output error $e(t) = y(t) - y_m(t)$ is the difference between the observed output $y(t)$ and the model output $y_m(t) = B(q^{-1};\theta)u(t)$.

FIR models are globally identifiable and are a special case of ARX-models ($A=I$). Recursive and direct methods for estimating the impulse response are discussed in Chapter 5.

ARMAX models

$$A(q^{-1};\theta)y(t) = B(q^{-1};\theta)u(t) + v(t)$$
$$D(q^{-1};\theta)v(t) = C(q^{-1};\theta)e(t) \qquad (2.6.3)$$

This type of model, with fully parametrised matrix polynomials, is estimated in the four-step IV-method, discussed in Chapter 5.
A form with a diagonal A-polynomial is the basis for a bootstrap IV-method.

General SISO and MISO models

A general model for SISO systems, cf. [Ljung, 1983]:

$$A(q^{-1};\theta)y(t) = \frac{B(q^{-1};\theta)}{F(q^{-1};\theta)}u(t) + \frac{C(q^{-1};\theta)}{D(q^{-1};\theta)}e(t) \qquad (2.6.4)$$

Recursive and iterative prediction error methods based on this model structure comprise many well known methods, such as Generalised Least Squares, Recursive Maximum Likelihood methods, the Extended Matrix Method, etc., see Ljung [1983].

The previous model can be extended to MISO systems by modelling the subsystems independently:

$$A(q^{-1};\theta)y(t) = \sum_{i=1}^{p} \frac{B_i(q^{-1};\theta)}{F_i(q^{-1};\theta)} u_i(t) + \frac{C(q^{-1};\theta)}{D(q^{-1};\theta)} e(t) \qquad (2.6.5)$$

In Chapter 5 we will discuss the output error variant, with $A \equiv I$. The output error model is globally identifiable if the polynomials B_i and F_i are coprime.

PEM's based on (2.6.4) and (2.6.5) have been implemented in several recursive and iterative methods in PRIMAL.

CHAPTER 3 A SCHEME FOR EXPERIMENTAL MODELLING

3.1 Introduction

In this section, we will consider conventional approaches to experimental modelling of process dynamics, before presenting, in the next section, a new, comprehensive scheme based on *interactive learning* in *real-time*, that is better suited to a systematic approach to experimental modelling in industrial projects.
Section 3.2 discusses the properties of this scheme, and Section 3.3 its different steps. Chapter 4 describes PRIMAL, a software environment that has been designed and built to effectively support the application of this scheme.

As described in the literature, the following steps are basic to the experimental modelling approach:

- Experiment design.
- Data acquisition (performing the experiment).
- Selection of a set of candidate models (the model set).
- Choice of identification techniques to select the best model from the model set.
- Model validation.

These steps are often presented in a scheme like the one shown in Figure 3.1.1, see also Ljung [1987], Söderström [1989], Isermann [1988].

An *experiment* is defined as the activity of generating a set of measurement data of process inputs and outputs during a certain time span. This definition includes measurements of the process during normal operation without using test signals.

Test signals are defined as signals that are superimposed upon the process inputs for the purpose of gaining information about the process dynamics. Test signals are usually carefully designed, so as to have a specific amplitude and frequency distribution.

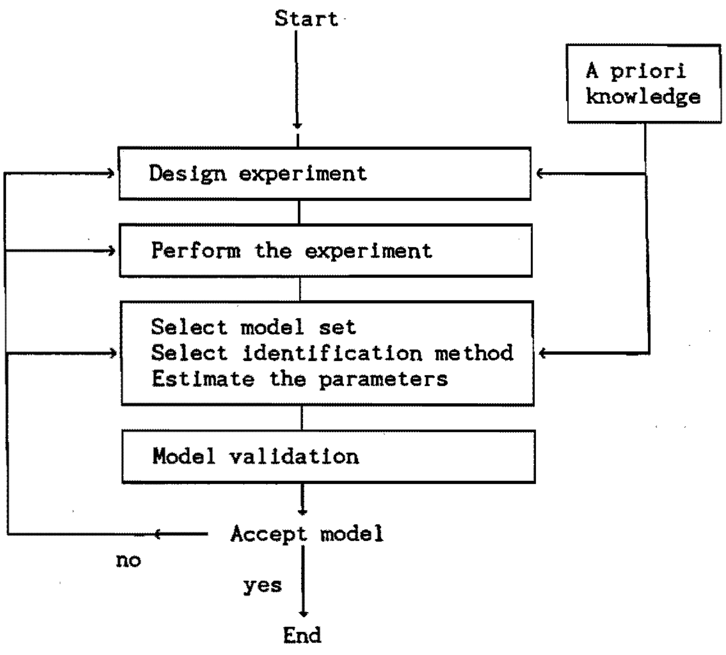


Figure 3.1.1:
The conventional identification scheme.

3.2 A new scheme for experimental modelling

Although the basic scheme presented in Section 3.1 is attractive because of its simplicity, in actual practice no project will start right away with the design of experiments for identification.
Instead, the first step is the definition of the project goals and the purpose of the required model(s). Different model purposes are, for instance, diagnosis, process monitoring, predicting future behaviour, and automatic control. Since the process may consist of several stages and show dynamical behaviour on widely different time scales, it is often necessary to break down the problem in several subproblems that have to be examined separately.

Once the need of having a model for solving a particular (sub)problem is established, one usually finds that there is insufficient information about the dynamical behaviour of the process to carry out identification experiments right away. In identification experiments, several decisions must be made, such as:

- which inputs and outputs are to be considered,
- the duration of the sampling interval,
- the length of experiments,
- the nature (e.g. type, amplitude and bandwidth) of the test signals.

To make a good choice and to perform an experiment that yields relevant information, the user must have first estimates of the process dynamics. This means that in a second step, before starting experiments, it is necessary to investigate the available knowledge about the process' behaviour. If this knowledge proves to be insufficient, exploratory experiments must be carried out first to acquire the missing information.

Further, it is usually necessary to connect additional equipment to the process, e.g. additional sensors and actuators or computer equipment for analysis of the measured data. After installation of such equipment, initial experiments and analyses may be carried out and these may gradually develop into the experiments needed for the estimation of adequate models.

The absence of sufficient a priori process knowledge to carry out informative identification experiments, leads to an interactive approach to experimental modelling. As the user gets results from the experiments and learns about the process, he may adapt the experimental conditions in order to improve the information content of the data. This leads to learning loops ("feedback loops") in the experimental modelling scheme.

We take the point of view that it is important, and often essential, that the analyses of the experiment data can be carried out on-line and in real-time.
The time available for experiments may be limited, in which case it is necessary to spend it carefully. No time should be lost due to wrong experiment design or process set-ups that yield unusable data. In off-line analysis this would be discovered only afterwards, whereas in on-line analysis the experiment may be monitored, and (recursive) analysis techniques may be used to verify whether the measured data show the required and expected properties. If these analyses show bad results, the experiment may be adapted. In this way the information content of the data can be improved as more knowledge of the process behaviour is built up. Generally, this leads to more accurate models (the model cannot be better than allowed by the data used to construct it) and/or shortens the duration of the experiments.

The collected data usually contain disturbance components, such as drift, measurement errors, and noise. As the experimental freedom of imposing test signals on the process inputs is usually restricted to low powers, these disturbances may have severe effects on the performance of the identification methods. Therefore, it is necessary to provide for a data conditioning step in the experimental modelling scheme, in which the "raw" experiment data are corrected for the undesired components as well as possible.

The design choices in the data conditioning, identification and model validation steps offer a large degree of freedom in finding a model. As yet there are no methods that are guaranteed to lead directly to a satisfactory model. Instead, the construction of a model is a learning process in which the user interactively tries out different model sets and identification techniques, using the acquired knowledge to adapt one or more of the previous steps. Learning is an essential part of each step (see Figure 3.2.1) as well as of the experimental modelling scheme as a whole (see Figure 3.2.2).

The basic form of a step is composed of three parts.
The first part represents the design choices. The second part represents the operations to generate results and the third part the interpretation of the results, which contributes to the knowledge of the process dynamics.

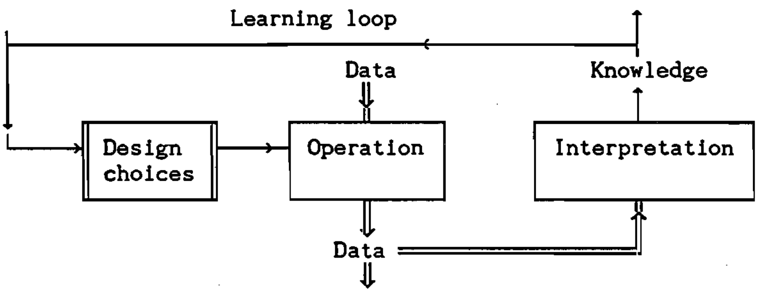


Figure 3.2.1:
The basic structure of a step.

Now we may distinguish the following collection of steps:

1. Definition of the goal and purpose of the model.
2. Investigation of available process knowledge.
3. Installation of equipment.
4. Experiment control and data acquisition.
5. Data conditioning and signal analyses.
6. Identification.
7. Model validation.

and if modelling is to serve control purposes:

8. Control requirements specification and control system design.
9. Control system testing.
10. Control system implementation.

The total scheme, shown in Figure 3.2.2, emphasizes that *interactive learning* and its associated loops are essential to the approach of experimental modelling.

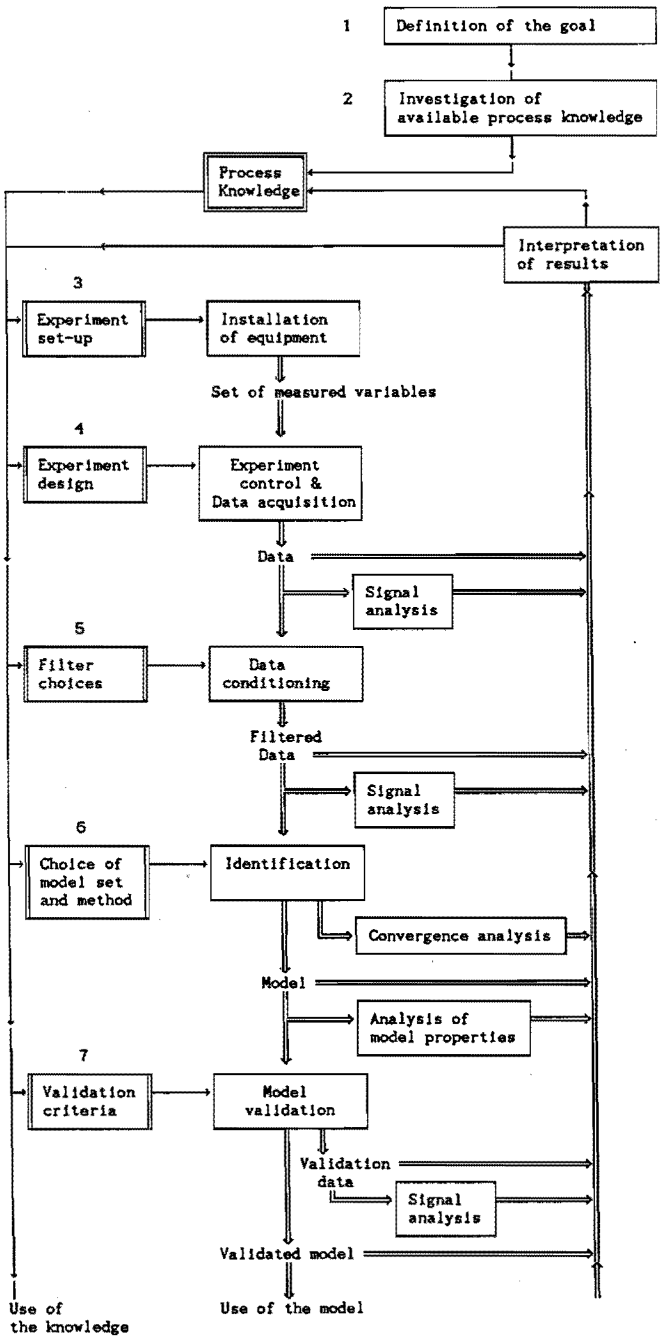


Figure 3.2.2:
The Experimental Modelling scheme (EM-scheme).

$\longrightarrow$ knowledge

$\Longrightarrow$ data, models, statistical results, model properties

□ actions, ▯ design choices

In the case of control system design the scheme is extended.

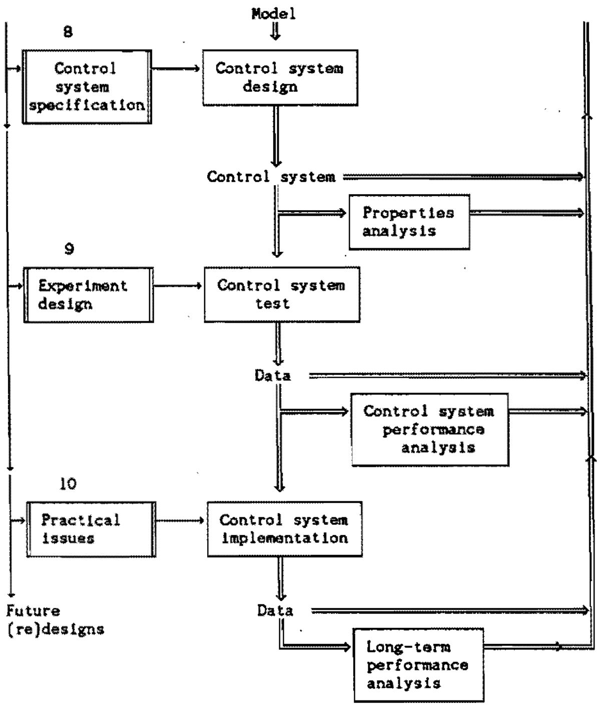


Figure 3.2.3:
Extension of the EM-scheme in the case of control system design.

Including control system design in the EM-scheme has some important advantages. Firstly, after simulation tests with the control scheme, the controller output can be superimposed on the process inputs as a special type of test signal.
Secondly, the approach used to estimate and verify models may be used equally well to test a designed control system's performance in actual plant operation. By using identification techniques the user can examine the nature of the deviations from the expected and required behaviour.

Comparing the EM-scheme presented here to the conventional scheme described in Section 3.1, we make the following remarks concerning our scheme:

a) It covers more completely and realistically the practice of experimental modelling and provides a better framework for applying the best theoretical methods.

b) From the start, the desirability of working in real-time is taken into account.

c) As a consequence, experiment design, experiment control, and data acquisition are essential parts of the scheme.

d) In order to improve the information content of the measured data, it is possible to carry out the analyses in real-time and to use their results for changing or adjusting the experiment instantaneously.

e) The scheme comprises a variety of learning loops.
Typically, the user works his way through the scheme in an interactive fashion, adapting the design choices in each step (and, if needed, in the previous steps) several times, before moving to a next step.

f) In practice, data conditioning proves to be an important step. As such, it does not get enough attention in literature.

g) The real-time approach allows the integration of process modelling with the design and validation of control systems.

To summarise, in practice experimental modelling of a dynamical process is best carried out as an interactive process in real-time, in which the user accumulates knowledge gained by each action, and uses it to adapt the experiment and the analyses in real-time.
The PRIMAL package has been conceived, designed and constructed to support this interactive approach, which has already proved particularly useful in the process industry (see Chapter 6).

3.3 Discussion of the experimental modelling scheme

Each step in the experimental modelling scheme consists of design choices, operations, results and their interpretation. In this section we consider aspects of each step and comment on the implementation of the scheme in PRIMAL.

3.3.1 Definition of the goal and investigation of the available knowledge

To apply the experimental modelling approach, the process problems must be translated into project goals: i.e. in terms of process behaviour requirements.

To design experiments for finding a suitable model and for designing a control strategy, it may be necessary to carry out a preliminary investigation of the available knowledge about the process behaviour. Its purpose is, among other things, to set up a list of process inputs and outputs, if possible, to draw a chart of interactions between the process variables, and to get impressions of the process dynamics.
The investigation must be carried out before installation of equipment and before the design of the first experiments. The choices to be made in those steps are to be based on the knowledge obtained in this preliminary investigation. The time required for installing equipment, initial experiment design and the initial experiments may easily exceed the time required for the identification with PRIMAL. The correction of wrong choices in the step discussed here may therefore cost much of the available time.

Because in practice every problem has its own unique aspects, it is difficult to set up a uniform approach, although this is exactly what we want. In the following an (inherently incomplete) list of points is presented that ought to be considered in the preliminary investigation of process behaviour. The list is presented here because, in my opinion, it is seldomly discussed in connection to experimental modelling in the process industries.
The ordering of the items does not imply any priority.

a) Inventory of the problems.
The investigation of the area of concern often leads to the conclusion that there are actually several problems that may, or may not, be completely unrelated. Their nature may be different (e.g. sensor/actuator errors, unsatisfactory control dynamics, process instability, start-up difficulties), they may relate to different parts of the plant, or, if arising in the same part, allow seperate treatment if they occur in other frequency bands. For the purpose of this thesis, it is presumed in what follows, that one such problem, that lends itself to a model-based solution, has been selected. All experiments carried out to solve that problem will be considered as one single "macro experiment", see Section 3.3.3.

b) Determination of the physical variables relevant to the process behaviour and, if possible, the construction of an interaction diagram.

c) Inventory of normal operating conditions and the restrictions concerning operation and safety.

d) Description of the instrumentation system: e.g. which quantities are measured, the location of the sensors and actuators, their range, speed, sampling rate, and accuracy.

e) Selection of the set of process inputs that may be used in control. These inputs should preferably have the properties (range, bandwidth) enabling them to compensate for the disturbances affecting the process.

f) Selection of process inputs that are measured, but can or may not be used as control inputs. Deviations caused by disturbances in these inputs may, conceivably, be compensated by feed-forward control.

g) Selection of the process outputs. These outputs must be direct or indirect measures of the process variables to be controlled. Often, indirect measurements have to be used instead of direct measurements, such as a temperature at a certain tray in a distillation column instead of a product composition.

h) Description of the existing control strategy. This addresses the automatic control loops as well as the character and frequency of the operator interventions.

i) Investigation whether there are relevant inputs and outputs that are not yet used by the control system.

j) Description of any known signal properties of the process inputs and outputs during normal operation, including, if possible, the unmeasured disturbances.
Important are the types of disturbances, their (statistical) properties, their possible origins and transmission paths.

k) Description of any known dynamical effects of each input on each output: initial estimates of the gain, number and range of time constant values, delays, signal to noise ratio, typical behavioural characteristics such as inverse response, instability, and oscillatory behaviour.

l) Investigation whether the problem may be solved with a better control system. From a controllability study it may follow that a better control strategy brings no significant improvement and that adaptation of the plant (in combination with an improved control strategy) is mandatory.

m) Investigation whether the types of tools, required to tackle the problems, are available.

n) Investigation of the (special) opportunities to carry out experiments.

o) Estimation of the time required for the experiments, or vice versa, whether enough reliable data may be acquired within the given time span.

The process knowledge to be acquired may be drawn from different sources, such as:

- Interviews with chemical engineers, control engineers, instrumentation engineers, the maintenance crew, and the operators.
- Literature on relevant physical and chemical knowledge.
- Available models, (theoretical, empirical, ..).
- Analysis of recorded data (e.g. off-line with PRIMAL).

After the preliminary investigation, it should be possible to decide upon the practicability of experimental modelling.

3.3.2 Experiment set-up and installation of equipment

To apply experimental modelling, a set of tools must be available to do experiment control, to generate datasets of sampled process inputs and outputs, and to perform real-time analyses. The way in which this may be realised depends on the available process instrumentation.
To avoid problems with redefining common terms, we introduce the term "Process Control System (PCS)" for any system that includes: analog data filtering, sampling, ADC and DAC, (low-level) control and computation facilities, and an operator interface.

We may distinguish three basic situations:

1. A plant without a PCS.
2. A plant equipped with a PCS, which may be a (usually heavily loaded) central control computer or a Distributed Control System (DCS).
3. A plant where powerful computing facilities are offered by the PCS, or are coupled real-time to the PCS.

Since a PCS performs time-critical tasks (sampling and control), it is generally not suited to the intensive computations involved in real-time analysis. Furthermore, the operating system, programming language, and human interface are usually designed for their specific tasks and do not provide the facilities required for interactive experimental analysis. Therefore, the analyses must usually be carried out on a separate computer that is directly coupled to the PCS. In the sequel this is called the "Analysis Computer".

Implementing the tools for experimental modelling on the Analysis Computer has the advantage that all steps in the EM-scheme can be performed largely independent of the PCS. A single software package may thus be developed for a wide range of different process control systems. Since the analysis computer does not have to perform PCS actions, it may be selected for its numerical processing, software development and interactive graphical input/output capabilities.

The installation requirements for the three cases mentioned above are:

ad 1. A PCS and an Analysis Computer must be coupled to the plant.
ad 2. An Analysis Computer must be coupled to the PCS.
ad 3. The tool-set may, conceivably, be implemented on the available computer facilities.

For a specific PCS, a program must be developed, that communicates with a program on the Analysis Computer, that is specifically designed for connecting to the PCS. The PCS program should provide the facilities to transmit measurement data to the Analysis Computer, and (optionally) to carry out experiment control actions, see Section 4.6. This set-up should offer sufficient flexibility for implementing links with widely different instrumentation systems.

Another important point is safety. The communication protocol is to be designed so that the operator remains in charge of the PCS and that possible failures of the Analysis Computer do not affect the PCS.

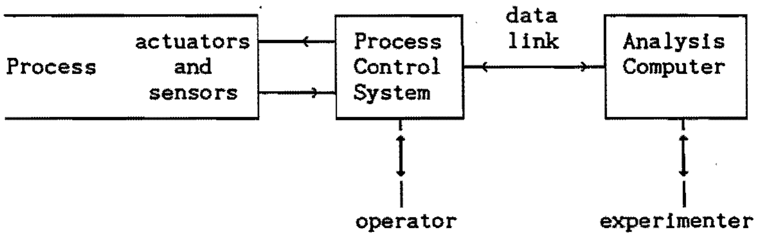


Figure 3.3.1:
The proposed computer configuration for cases 1 and 2.

Principal design choices are:

a) The selection from among the available sensors and actuators.
b) The installation of additional sensors (and actuators).
c) The choice of the computer configuration and data link.
The computer's performance, data link bandwidth, and speed of the data storage influence the attainable sample frequency and number of signals that can be handled simultaneously.

3.3.3 Experiment design and experiment control

In parallel to the installation of equipment we may start to design experiments. Here we have the following design choices:

- Selection of inputs and outputs from among those available for analysis.
- Choice of sampling interval(s).

- Choice of experiment duration.
- Choice between SIMO or MIMO experiments.
- Choice of test signals.

The selection of inputs and outputs has been discussed in Section 3.3.1 and is usually done on the basis of a priori knowledge.

The selection of inputs is critical. Missing an important measurable input may result in a model of low quality.
Therefore, its is generally advisable to take in as many signals as is technically feasible. This offers the opportunity to examine the correlation between inputs and outputs at a later time. A significant advantage of the on-line approach is the opportunity to perform experiments with test signals in these inputs.

Experiments

We find it useful to make a distinction between "explorative experiments", "identification experiments" and "application experiments".

Explorative experiments

The explorative experiments are design-oriented, i.e. they are used to make the best possible design choices for the identification experiments.

Preliminary experiments of this type are used to characterise the normal process behaviour, exploring the amplitudes and spectra of inputs and outputs under normal operating conditions and the nature and spectra of the disturbances. Correlation analysis may be used to verify and investigate process interrelations.

Test signals may be used to test assumptions to be made in the identification experiments about linearity, stationarity and causality, and to get initial estimates of the values of the gains, delays, and time constants.

Backx [1987, 1989] describes experiments for this phase. One of these experiments involves a stair-case signal to investigate linearity, hysteresis, gains, and time constants. Although such signals may appear to be well-suited for exploring the process, this experiment is not very appropriate if it changes the operating conditions for a long time interval; also it is time-consuming, and the results are poor when a significant trend is present.
A better experiment is to use a three-level signal (or a PRBS) starting with small amplitudes and increasing/decreasing the amplitude in a number of stages.
The advantage is that:
- The change in the operating conditions may be kept smaller.
- The static gain is estimated correctly if sufficient energy is put at frequencies in the pass band of the process.
- The procedure is less sensitive to drift.
- More accurate information about the time constants may be obtained.

The experiment is, however, less suited to explore hysteresis.

Identification experiments

Identification experiments are intended to find dynamical models of the process.

Generally the data generated by a process in normal operation is not well suited for identification, because the inputs may not contain sufficient information at all frequencies of interest. Therefore, it is desirable to superimpose carefully-designed test signals upon the process inputs. If little knowledge about the dynamics is available, these signals should excite the process in a broad frequency band. Depending on the frequency band of interest, the test signals may be applied to an actuator, a controller set point, a controlled signal or ad-hoc "actuators" (e.g. an injection of tracer material).

Test signal types that are widely used in practice include periodic signals, such as block signals and sums of sinusoids, and aperiodic signals such as pulses, steps, and noise signals. In several of the identification experiments carried out with PRIMAL, the PRBS has been exploited because of its nice properties:

1) The spectrum of a PRBS with a clock period of λT (with λT some positive integer multiple of the sampling interval T), amplitude α and periodicity M may be approximated if M is large by:

$$\Phi(\omega) = \frac{2\alpha^2}{\lambda\omega^2 T}(1 - \cos(\omega\lambda T)) \qquad (3.3.1)$$

   This spectrum has zero's at all integer multiples of angular frequency $\omega_0 = 2\pi(\lambda T)^{-1}$. Thus, the PRBS excites the process at all discrete frequencies up to ω_0. (persistently exciting).
   The design variable λ may be used to shift the energy content of the PRBS to match (slightly exceed) the bandwidth of the process. The choice of α is a trade-off between the maximum allowable input and output variations and the length of the experiment.
2) The fixed amplitude and zero average value facilitate its acceptance in practice.
3) A PRBS can be easily generated using shift registers, cf. [Godfrey, 1969; Davies, 1970]
4) Mutually independent PRBS may be generated by selecting proper initial seeds in a maximum length PRBS with a period $M >> N$ (the experiment duration).

Remark:
A PRBS with a clock frequency below the sampling frequency can also be generated from a two-level white noise by decreasing the probability of change-overs. The advantage of this approach is that no energy is put in side-lobes, cf. [Tulleken, 1988].

Remark:
The design of optimal experiments, based on detailed process knowledge, and taking into account the intended use of the model, [Gevers, 1986; Goodwin & Payne, 1977; Mehra, 1974] has not yet been fully exploited in this thesis.

Application experiments

When models have been formed and/or control systems have been designed, it is desirable to use and verify their performance in actual practice. Therefore, experiments may be carried out, like:

- "Running" a model in parallel to and synchronised with the process, predicting future process behaviour and comparing it to the actual behaviour.

- Implementing a control system on the Analysis Computer or PCS and assessing its performance, using the methods of the EM-scheme suitable for closed loop systems.

Like the explorative and identification experiments, these experiments are part of the learning process.

A macro experiment

During the available time, the experiments described above may be carried out repeatedly. Preferably, they should be part of one single <u>macro experiment</u>. In this way, data conditioning can be improved and the start-up effects in each analysis may be avoided. Moreover, the often difficult choice of selecting the total number of samples N can be relaxed to specifying an upper limit only. The length of the individual experiments may be taken on the basis of intermediate results.
In what follows the macro experiment is often referred to as *the* experiment.

It is advisable to take the sampling frequency a factor 5 to 10 higher than needed on the basis of the time constant values of interest. This will improve data conditioning, see Section 3.3.4.

<u>3.3.4 Data conditioning</u>

Generally, the raw measurement data is not well suited to identification. It may suffer from outliers and trends, and the dynamical relations may be altered by sensing delays and non-linearities.
Practical experience showed that proper conditioning of the data may have a decisive influence on the attainable quality of the model in the identification step. Therefore, it is better to examine the raw data and prefilter it, applying one or more of the following filtering steps, instead of accommodating the identification methods to more or less remedy some of the problems.

Data conditioning comprises:

1. Correction for known fixed delays.
2. Correction for known (static) non-linearities.
3. Outlier correction and other signal repair.
4. Trend correction.
5. Noise reduction.
6. Data reduction.
7. Offset correction and scaling.

The techniques for examining the data include visual inspection, signal analyses, and identification. The importance of off-line data conditioning has been emphasized by Backx [1987].

Correction for known fixed delays

If one or more inputs to the model are measured with a significant delay, correction is desirable to prevent apparent non-causal response of the outputs to the inputs.

Furthermore, delay correction is useful if the dynamic response of the output to an input is delayed by a large, fixed time interval, for instance due to transportation time or (chemical) analysis. Most identification techniques offer no means of simultaneously estimating the delays and the other process dynamics. Small delays can be incorporated by increasing the model order.
In a MIMO system not all delays can be compensated for if $p*q$ is larger than $p+q-1$.

Estimates of the delays may be obtained by correlation analysis, by direct impulse response estimation, and from physical process knowledge. Better techniques, which also work for short datasets and noisy data, will be introduced in Section 5.5.

Correction for (static) non-linearities

Correction for non-linearities is possible when there is a known non-linear behaviour, such as non-linear actuator and/or sensor characteristics.

Signal repair

Measurement data may be affected by sensor failures and outliers, whose energy content may have a substantial influence on the identification.
Failures in the data sequence that are easy to detect, may be eliminated before further processing the data.

Automatic correction of outliers is implemented in the data conditioning step by simple detection techniques.

As described by Backx [1987], the simplest technique is to compute the signal mean $\bar{x}$ and standard deviation $\sigma(x)$ and to detect outliers with:

$$|x(t) - \bar{x}| > S.\sigma(x) \tag{3.3.2}$$

where S is the "shaving strength".
For signals showing drift, this technique may not be expected to work well, since the detection boundary does not change as a function of the time t.
An approach to this problem is to filter x(t) with a high-pass filter L(q) and to use the test:

$$| L(q)x(t) | > S.\sigma(L(q)x(t)) \tag{3.3.3}$$

Alternatives to these tests may be employed, that use the median and/or use information about the amplitude distribution of x(t).
Unlike the mean, the median is not sensitive to the amplitude of the outliers.

Detected outliers may be replaced with interpolated or mean signal values. Note that direct filtering of the data with a low-pass filter also reduces the outliers, but does not affect their low-frequency contributions. As a consequence, the influence of the outliers spreads to neighbouring samples.
Outliers that are not detected in this stage may be found later in the identification step, showing up as large residuals. In such a case the repair of the data should be reconsidered.
In case I of Chapter 6 the use of outlier correction is demonstrated.

Remarks:

a) By estimating the mean, median and variance recursively, the outlier detection methods are suited to operation in real-time.

b) Ljung [1987] suggests to "robustify" the prediction error methods for the occurrence of outliers by reducing the weight of large prediction errors in the criterion function. The robustification requires the setting of some ad-hoc boundary on the prediction error. However, since outliers also disturb other data conditioning steps and subsequent signal analysis, we prefer to remove them beforehand.

c) Although the automatic techniques for outlier detection may work quite well, practice shows that the human inspection of graphically presented data is of great value. The human eye proves to be a superb "filter" for detecting strange signal behaviour.

Trend correction

If low-frequency disturbances (trend), like drift and periodic variations, are present in the data and have a high energy content

compared to the frequency band of interest, they may severely influence the parameter estimates.
Trend and the frequency band of interest should therefore be separated. A suitable approach is to separate the trend by a high-pass filter, before estimating the process dynamics.
The selection of a cut-point frequency between trend and signal is usually guided by the intended use of the model. Removing the trend discards all information on the low-frequency behaviour and thereby reduces the accuracy of the static gain estimate.
Alternative approaches to deal with the trend have been suggested by e.g. [Isermann, 1974; Baur, 1976], involving the estimation of the parameters of a trend model in parallel to the parameters of the process model.

Trend filtering should be done with a sharp filter. Low-order filters may reduce the information content in the frequency band of interest and/or not sufficiently eliminate the trend in the data.
The impulse response of an ideal high-pass filter may be approximated by a FIR-filter, [Jackson, 1986; Stanley, 1975].
The desired frequency response $H(\omega)$ is expanded into a Fourier series. The impulse response $h(t)$ of a FIR-filter is constructed by truncating the Fourier series and applying the inverse z-transform. This will result in a filter showing some pass band and stop band ripple. The ripple may be reduced by multiplying $h(t)$ with a suitable windowing function - eg. a Hanning, Hamming or Blackman window - at the cost of a wider transition band.

Filtering the data by convolution with $h(t)$ introduces a linear phase shift, which may be eliminated centering $h(t)$ around zero, i.e. applying a non-causal filter.

$$x_{tr}(t) = \sum_{\iota=-M}^{M} h(\iota)x(t+\iota) \qquad (3.3.4)$$

Remarks:

a) The FIR-filter introduces transient effects at both ends of the data sequence. The initial values for the filter must therefore be chosen carefully.
In the on-line version the linear phase shift of the filter introduces a delay of M/2 samples in the filtered signals.

b) To avoid unnecessary complications, the measured process inputs and outputs should preferably be filtered with identical filters.

c) In the SISO case, filtering the data is equivalent to introducing a frequency weighting in the criterion function of a prediction error method.
Filtering input and output with $F(q)$ is equivalent to a PEM applied to the unfiltered data with a prediction error weight:

$$| F(e^{\iota\omega}) |^2 \, \Phi_{\varepsilon}(\omega;\theta). \qquad (3.3.5)$$

Data reduction

If technically feasible, it is advisable to sample the sensor signals at a higher sampling rate than required for the estimation of the dynamics. A pragmatic reason for this approach is that the bandwidth of the process may have been estimated incorrectly from the a priori knowledge. While it is possible to reduce the sampling rate for identification by data reduction, the opposite is impossible.
Furthermore, the excess data may be used in signal repair, trend correction, and noise reduction, before reducing the data.

Data reduction should take place by filtering out all frequencies higher than the new sampling rate (to prevent aliasing effects) and subsequently decimating the data.
If data reduction is not performed, we may expect problems in the estimation of the low-frequency behaviour of the process, especially when employing equation error techniques.
Moreover, since the signal values change little from one sample to the next, we may expect numerical problems with the inversion of the data matrices in the identification and the use of the model, as all poles will cluster near $z=1$.

Offset correction and scaling

Offset correction is desirable to avoid that it must be explicitly accounted for, e.g. by including the offset as an additional parameter in the parameter vector, or by estimating a model on differenced data (ARIMA-models).

If the values of different signals differ by several orders of magnitude, it is necessary to scale the data to a matching numerical range in order to avoid numerical problems due to ill-conditioned matrices.

3.3.5 Identification

In the context of this thesis, the estimation of the dynamical behaviour between inputs and outputs is done using black box techniques.

The design choices in black box identification include:
- Model representations, (state space, MFD, etc.).
- Model orders and parametrisation (of the transfer function and the noise model).
- A criterion (PEM's) or correlation vectors (IV-methods)
- A numerical procedure.

Based upon physical insight, the intended use of the models and the analysis of correlation functions and spectra, it is possible to get an impression of suitable model orders and model parametrisations.

If this impression is not very clear, it is desirable to apply specific methods for model order and model structure selection. Although much work concerning this subject is reported in literature, [Bohlin 1987, Stoica 1986, Janssen 1989, Söderström 1989, ..] there seem to be no established general methods for the approximate modelling case. The decision requires subjective judgement, [Ljung, 1987, Ch. 16, Söderström, 1989, Ch. 14], taking into account the intended use of the model.

A practical approach is therefore to apply several identification methods, testing different model orders and parametrisations, and compare their performance using model validation and model utilization techniques.
In accordance with our strategy of interactive learning, the user should be provided with appropriate facilities for making choices from among a variety of methods, based on different model parametrisations and estimation techniques, instead of relying on a single technique. The user should further be enabled to activate several identification methods in parallel and, if possible, apply them to the real-time data.

Identification methods may be divided into several classes, depending on the model parametrisation and on the numerical techniques used. The methods implemented thus far in PRIMAL comprise recursive and iterative prediction error methods for equation error and output error type models, and bootstrap and four-step instrumental variable methods, see Chapter 5.

The goal has been to first implement the most promising black box techniques discussed in the literature. Based on practical experience with these methods, those that appeared to perform best have been selected and effective new variants have been developed.

3.3.6 Model validation

In the model validation step we investigate whether the model is in sufficiently good agreement with the data and whether it is suited for its intended use. Another question is whether better models may be found, showing a better or comparable fit while using fewer parameters.

We shall briefly discuss two classes of tests. The first class confronts the model with the a priori knowledge and the data, preferably a set of data that has not been used in the identification. The second class compares properties of the models obtained with different identification methods, model orders or model parametrisations; these tests may also be used in model structure selection.

In the first class of tests, a model may be matched with a priori knowledge by inspecting its input-output behaviour, e.g. its impulse response and frequency response.
A test that stresses the low-frequency predictive quality of the model, is the simulation performance. The model response to the

process inputs is computed and the output errors, i.e. the differences between the observed process outputs and the model outputs are determined.
The behaviour of the output error can be examined by e.g. visual inspection, correlation analysis and spectral analysis.

The output error test has the advantage that it validates the estimated process model and does not rely on assumptions concerning the noise model. It may also be used for validation of the model on another dataset (cross-validation).

By computing the cross-correlation functions between the inputs and the output errors, applying confidence limits for independency, it can be checked whether all contributions of the inputs to the outputs have been captured by the model. The computation of these confidence limits relies on a variant of the central limit theorem, cf. [Ljung, 1977]. If e(t) and u(t) are independent, then:

$$\sqrt{N}\, R_{\epsilon u}(\tau) \in As\mathcal{N}(0,P) \qquad P = \sum_{k=-\infty}^{\infty} R_{\epsilon}(k)R_{u}(k) \qquad (3.3.6)$$

Correlation for $\tau>0$ indicates that the modelling has not completely succeeded and may be further improved.
A comparable type of test, applicable to prediction error methods, is a "whiteness test" of the residual $\epsilon(t)$.

In the second class of tests, the validity of a model is assessed by comparing it to other estimated models.
Since the various identification methods use different model structures, the comparison of parameter values is impractical. Moreover, the parameters were considered to be just vehicles in the estimation procedure.
A class of techniques that depends on penalising model complexity, (e.g. Akaike's AIC-criterion, the Final Prediction Error (FPE) criterion, and hypothesis testing techniques) apply to hierarchically ordered model sets and are therefore usually restricted to a certain identification method. Several such criteria have been compared by Freeman [1985] and are used in the identification methods in PRIMAL.

In order to compare different identification methods, it is necessary to use tests that are independent of the particular parametrisation and take the intended use of the model into account. Such tests should involve only the input-output behaviour of the process model. Along the lines of the interactive learning strategy, this may be done by (graphically) comparing the impulse responses, Bode or Nyquist plots, and output error behaviour of different models.

A useful quantity to compare simulation performance is the mean square relative output error:

$$mre = \sum_{t=1}^{N} \{\| y(t)-ym(t) \|_2 / \|y(t)\|_2\} \qquad (3.3.7)$$

$$ym(t) = G(q^{-1};\hat{\theta})u(t) \qquad (3.3.8)$$

The relative output error of individual outputs is defined as:

$$re_i := \sum_{t=1}^{N} [y_i(t)-ym_i(t)]^2 / \sum_{t=1}^{N} y_i^2(t) \qquad i = 1..q \qquad (3.3.9)$$

3.4 Conclusions

In this chapter we have developed a comprehensive strategy for experimental modelling that is, in contrast to conventional approaches, based on *interactive learning*, *real-time analyses* and *adapting the (choice of) experiments* on the basis of the acquired knowledge.

This strategy has led to an experimental modelling scheme comprising various steps and learning loops, and permitting user interaction, especially in the often time-consuming early phases of a project when knowledge about the process behaviour is limited.
Important advantages of on-line (real-time) experimental modelling over off-line analyses are the opportunities to:

- Improve the experiments on the basis of intermediate results of the analyses, leading to better results and/or a shorter project duration.
- Include the testing and validation of control systems in the EM-scheme.

Experience with the experimental modelling scheme in industry, has led to a consideration of issues like, how to:
a) deal with a priori information.
b) connect to the process.
c) plan experiments and deal with the intermediate results.
d) condition the raw measurement data properly.

To follow the strategy depicted in this chapter, an environment that support it is needed. The package we have constructed for this purpose: PRIMAL, is presented in the next chapter.

CHAPTER 4 THE PRIMAL PACKAGE

To enable a user to follow the strategy described in the preceding chapters, it is necessary to provide a coherent collection of tools that assist him in each step of the scheme.
Since most steps involve numerical computations based on experimental data, it is natural to provide the tools as part of a computer package implementing the scheme.
The development of such a package, suitable for application in industrial practice, has been one of the principal aims of the work described in this thesis.
The package, with the name PRIMAL: "Package for Real-time Interactive Modelling, Analyses and Learning" has been developed as a result of this work. Its main objective is to support users in developing mathematical models of the behaviour of industrial processes and in exploiting such models, for example in the design of improved control strategies.

How the experimental modelling scheme and the demands and wishes of the intended users are translated into requirements with respect to the package, is discussed in Section 4.1. Section 4.2 treats the user requirements and Section 4.3 the software requirements. In some cases the same requirements result from different motives.
Section 4.4 discusses the structure of the package and how it meets the requirements. Section 4.5 deals with the software organisation and the implementation of the basic mechanisms. Section 4.6 treats the implementation of experiment control in more detail. Section 4.7 lists a sample session and discusses some important user aspects.

The PRIMAL project started in 1979 and entered a new phase in 1984. By then, no software tools were commercially available for (professional) use in industrial practice that covered experiment design, data acquisition, real-time signal analyses, real-time identification, control design and control system implementation. For this reason, an environment has been designed and constructed with the capability to serve both as a professional tool for use in industry, as well as a development environment for methods of data analyses.
Since then, other packages addressing the experimental modelling problem have emerged. A comparison with PRIMAL will be made in Section 4.8.

4.1 Package requirements

The Experimental Modelling scheme, as depicted in Figure 3.2.2, has a number of important characteristics to be translated into requirements:

- Experiment design, experiment control, data acquisition, and model utilization, e.g. for testing control systems, are essential parts of the scheme.
- Each step in the scheme represents a set of design choices and (computational) methods. To obtain a model it is necessary that appropriate methods are provided for all steps in the scheme.
- Experimental modelling is not carried out in a single sequence of steps, but instead the user works his way through the scheme several times interactively, each time adding knowledge (learning) about the dynamics.
- It is essential that analyses on the experiment data can be carried out while the experiment is running. On the basis of the results, the ongoing experiment may be adjusted in order to improve its information content.

From these characteristics, requirements are derived, as discussed below.

4.1.1 Experiments and data acquisition

The package must fully support experiment design, experiment control and data acquisition. Consequently, it must have tools to establish a physical link with the Process Control System (PCS), as defined in Chapter 3. Measured data, test signals, experiment control messages and controller parameters must be passed through this link in real-time.
The user must have facilities to interact with the running experiment and at any time the package must be ready to promptly respond to his commands.

To make on-line changes feasible, the user must have facilities to inspect and process the incoming data.
The package must be equally suited to off-line processing of data gathered previously, or measured by a (remote) facility that does not allow on-line coupling.

4.1.2 Full functionality

For each step in the EM-scheme, appropriate methods must be provided by the package. This means that methods must be available for experiment design, experiment control, data acquisition, data conditioning, signal analysis, identification, model validation, as well as for control system designing, testing and validation.

Therefore, the package must provide a suitable environment for embedding the methods, in a framework that provides:
- Fast and easy initiation, modification and termination of a method.
- Access to the output data of a method by any other method.
- Full database support.

4.1.3 Interactive learning

A quintessence of the EM-scheme is that, at any moment, the user must be able to make a free choice among all facilities the package provides, not being hindered by any preconceived path set out by the designer. This requires, first of all, a flexible user interface.

The results of the methods must be presented in a format that directly adds to the understanding of the process. In addition to presenting models as sets of matrices, which may be difficult to interpret, particularly by plant people, the methods must present user-readable results, like time responses, Bode plots, polar plots, etc.

The package does not yet include an expert system, because it is difficult and maybe up till now impossible to formulate a satisfactory set of rules for experiment design, data conditioning, and modelling of (complicated) multivariable systems. Instead, the package supplies interactive methods and powerful presentation tools, and is provided with an on-line help system and a complete set of comprehensive manuals [Van der Linden & Renes, 1989a-e].

4.1.4 Real-time aspects

To improve the information content of the measured data and, as a result, the quality of the estimated models, it should be possible to analyse the measured data while the experiment is running. This implies that analyses may operate on the already present intermediate (updating) set of data and may *synchronise* with the data still coming in from the process.
This requirement may not be restricted to a single analysis method. Several analyses should be able to operate on the same data in parallel to each other. Extending this concept, it should further be possible to apply *any* method to the *intermediate* results of *any* other method provided, of course, this makes sense.

This has several important consequences.
Since each method may operate upon the intermediate results of other methods, methods may be *chained*. For instance, the intermediate results of analyses may be chained to the graphical module, presenting them together with the updating experiment data. User inspection of intermediate results is especially useful for recursive or iterative methods and may lead to interactive modification of the experiment or the analyses.

By chaining methods, the complete EM-scheme may be realised out while the experiment is being performed.

Since the user does not have to wait with next commands until a method is finished, he may start-up several analyses in parallel, which is important in both on-line and off-line applications.

In on-line application, several analyses may be started up on a (slow) process and the updating measurement data and intermediate results of several analyses may be presented simultaneously.

In off-line application, the user may try out several methods in parallel without the need to wait until a method has finished. He may start at once to investigate intermediate results.

Because methods that work on-line may be active for the duration of the complete experiment, it is necessary that the parameters affecting their activity and computations may be changed interactively at any time by the user. The package should therefore provide appropriate facilities to this purpose.

Finally, it must be possible to start-up a method so that it first processes already collected data and then, as it catches up with the incoming measurement data, synchronises with the experiment.

4.2 User requirements

4.2.1 Application in industrial practice

For use in industrial practice the package must meet additional requirements. Most importantly, a package coupled to an industrial process must be absolutely safe to use. Factory upsets caused by errors in the package or erroneous user inputs are unpermissible.
This requirement has important implications for the design of the experiment control and for the software design, which will be treated in Section 4.4.
All actions by the package must always remain within preset limits, defined during experiment design. User input not conforming with these limits or erroneous input must be refused. The package may not interfere with essential PCS supervisory tasks or the alarm system. The PCS maintains full control of the process. User interaction with the process through the package must be checked automatically.

For the analysis of the measured data and the evaluation of model quality, the experiment conditions should be reconstructable. This implies that the experiment must be completely defined and an automatic log book must be kept that stores and time-stamps all user actions, experiment control messages and, where possible, operator actions. This requirement is extended to the analyses. Thus, all output data and user-adaptable parameters of the methods must be stored in the database, so as to make it easy to reconstruct the experimental situation and to reproduce results at any later time.

Should the PRIMAL system fail, due to software errors or hardware failure, the process may not be disturbed and the database contents may not be lost and must remain consistent.

The way the requirements are met will be discussed in Section 4.6.

4.2.2 The user interface

The package is intended for users, either experienced or inexperienced in the field of systems and control theory and matrix algebra. Utilisation of the methods must be possible with marginal knowledge of the underlying theory and the output must allow direct interpretation by the user. Also, a user may be familiar or unfamiliar with the user interface or certain uses of it.

For the interaction with the user, various user interface models may be employed, which differ in the degree and manner of guidance given to the user. Experienced users demand high interaction speed and flexibility. Inexperienced users are, above all, interested in clear guidance. Both categories must be served by the package.
The following requirements have been identified:

a) The user interface must be easy to learn for users unfamiliar with the package. Inexperienced users must be provided with effective guidance.
b) Experienced users must be able to use the interface with maximum flexibility.
c) All methods must present themselves to the user in a standard manner, i.e. a standard 'look-and-feel'.
d) The user interface must be consistent.
This means that the interaction between user and package always follows the same predictable and easy to understand patterns, and that exceptions to the apparent logic, which are confusing, are avoided.
e) The user interface must be robust, i.e. wrong input may never lead to package failure.
f) The check for erroneous and incompatible parameters must be carried out at parameter input. Error reporting, error recovery and user assistance must be provided on-line.
g) Output of a method must be presented in an understandable form, enabling direct interpretation.
h) Since the package deals with analyses on generally large amounts of data, powerful presentation facilities must be provided, that are capable of presenting intermediate data of the methods.
i) The package must provide reasonable default values for all user-adaptable parameters of a method.

4.3 Software design requirements

In this section additional requirements are defined, that have no direct relation with the EM-scheme, but pertain to the software and to the manageability of an extensive software development project.

Package safety

For the package to be accepted in an industrial environment, several important requirements must be met.

Most importantly, a package coupled to an industrial process must be absolutely safe. The experiment control aspects of this requirement have already been discussed in Section 4.2.1.
The development of bug-free software poses a problem that is difficult, if not impossible, to solve, especially regarding software with real-time and (pseudo-)concurrency aspects.
However, the problem may be alleviated by the following set of requirements concerning software structure and software development methods:

a) The software must be partitioned into small, independent modules in such a way, that a bug in a module will only affect that single module. Error propagation between modules must be avoided. If we follow this strategy, the demand for absolute safety may be restricted to the Experiment Control Module, which takes care of all interactions between the package and the PCS.

b) Every module must be carefully divided into functional layers, to prevent or reduce error propagation inside a module. As a result, testing will be simplified, because individual layers can be tested independently.

c) Rigorous programming, testing, and documentation standards must be enforced upon each individual software designer involved in the project.

Multi-user support

The package must support several users simultaneously and each should be able to generate and manage his own data, without being confronted with the other users. However, a user should also be able to access data generated by others.

PRIMAL packages implemented on different computers must be able to communicate, so that the users may access the databases of users on other computers.

Portability

The package must be able to run on different computer systems, with different hardware environments and different operating systems.
This results in the following requirements:

a) A widespread (high-level) programming language, available on many computer systems (in industry).
b) Operating-system dependent procedures, that cannot be avoided, must be concentrated in a single layer. This layer will be called the "Virtual Operating System".

c) Device-dependent procedures must be concentrated in a single "Device Driver" layer. Constructing and changing device drivers must be straightforward.

Extensibility

The package must provide an environment in which new methods may be implemented easily. This thesis restricts itself to experimental modelling and control, with an emphasis on black box methods. The package design must also allow for other (user-added) methods.

A suitable software design environment should support rapid prototyping of new modules and easy incorporation into the package.

Interfacing with other software packages

For modelling and control, various software products are available, with specific capabilities like simulation, identification and/or control design. PRIMAL must be designed as an "open" system which allows for transfer of data and commands to and from other software products.

PRIMAL must be able to communicate with a diversity of Process Control Systems. This requires a flexible process interface, since the available hardware capabilities, programming tools and performances of these systems vary widely. The structure of the process interface is discussed in Section 4.6.

Software management and maintenance

For a software design project that stretches over several years and combines the contributions of several software designers, including relatively inexperienced students, it is necessary to:

a) Start-out with a clearly-defined set of requirements and a rigorous, but flexible, design concept.
b) Divide the project in several design phases, according to a software development scheme, including software maintenance and testing.
c) Define different functional layers and use a modular design strategy.
d) Set strict rules for structure, documentation and lay-out of the software.

4.4 Description of the design

This section describes the design of the PRIMAL package. The first part introduces the basic concepts and the structure of the package. The second part discusses the package's properties and the realisation of the requirements listed in the previous sections.

4.4.1 Terminology

Data object

A data object is a set of one, two or three dimensional matrices with numerical or text elements. Data objects are identified by their name.

In principle, each data object is structured, i.e. it contains several matrices that belong together in some sense, for instance a set of matrices constituting a state-space model, or a set of simultaneously measured signals.
Structured data objects may contain other structured data objects.

Dataset

A dataset is a structured data object that is stored and registered in the PRIMAL Database.

The PRIMAL Database

The PRIMAL Database is a hierarchically structured collection of datasets.

Dynamical datasets

A dataset is dynamical if its contents are subject to change. In other words: new data objects may be added to the dataset and data objects already present may be altered. After closing the dataset to changes, it is called static.

Modules

A module is a component of the PRIMAL package that performs an operation on a set of data objects and produces a set of new data objects as output.
To stress the difference between data source and destination, we will use the term *input data object* for data used as input to a module and *output data object* for data produced by a module.
The input data objects are read, and the output data objects are created, filled (written), and updated by the module.

The Monitor

The Monitor is the component of the PRIMAL package that implements the central user interface, module management and Database management.

Messages

A message is a collection of data objects that is sent from the Monitor to a module or vice versa.

Parameters

Parameters are data objects that are transferred from the Monitor to a module, or vice versa, as part of messages.

4.4.2 Package design

The PRIMAL package consists of the following principal components, cf. Figure 4.4.1:

- The Monitor
- Modules
- The PRIMAL Database

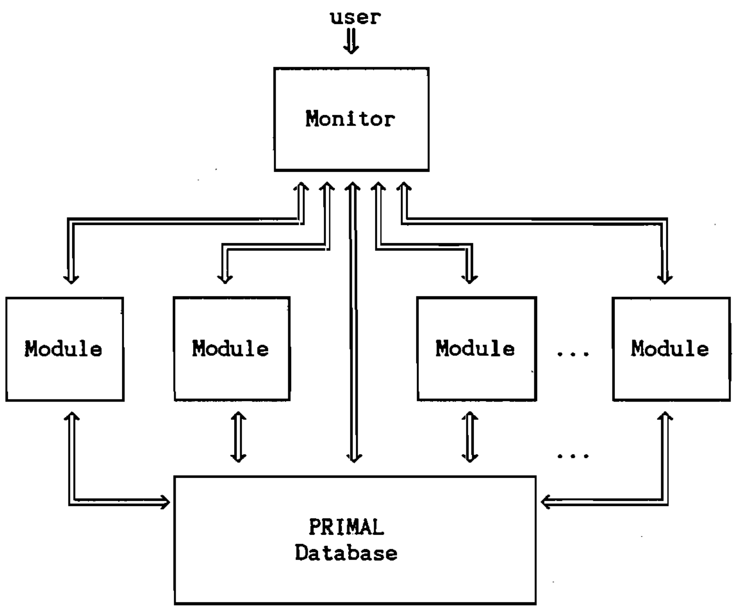


Figure 4.4.1:
The three principal components of the PRIMAL package.
The arrows represent the transport of data objects.

Modules

Each method, be it for experiment design, experiment control, data acquisition, signal analyses, identification, control system design, etc., is implemented as a module, that takes data objects as input, performs an appropriate operation and stores the results in output data objects. For instance, correlation analysis takes a set of signals as input data object and, after computation, produces a set of correlation functions as an output data object.

A module can get data from various sources and direct its output to several destinations, see Figure 4.4.2.

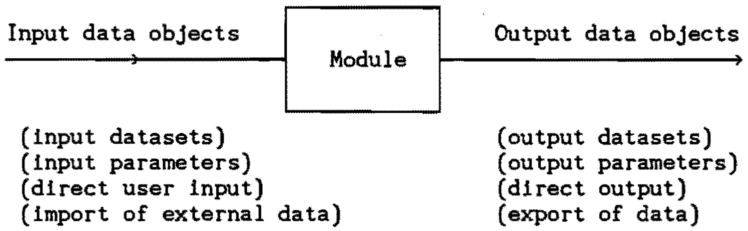


Figure 4.4.2:
Module input and output.

Parameter input and *direct user input* are the normal mechanisms for initialising and adjusting the variables that determine the module's actions, while as a rule *datasets* are used as the main source of data objects. Exceptions to this rule are modules for generating datasets that take their input from the user and do not require input datasets.
Most modules store the results of their computations in output datasets. In contrast to this rule, the presentation modules produce no datasets, but use *direct output* to plot or list data objects on a screen or on hard-copy devices.
Parameter output is used to send status, error and warning messages to the Monitor.

A module may import data from external sources or export data to external destinations, such as files of other packages. This type of i/o is local to the module and is not visible to the Monitor and other modules. The products of the data export are not stored in the database.

PRIMAL modules possess the following properties:

a) A module may execute in parallel to and concurrently to other modules. Thus, several modules may be active at the same time.

b) The input to a module may come from input datasets, input parameters, external sources, and directly from the user.

c) Module output is transferred to output datasets, output parameters, external destinations and can be directed to output devices for presentation, cf. Figure 4.4.2.

d) Any module is independent of all other modules, except for the synchronisation of the data transport through dynamical datasets.

e) A module may take its input from dynamical input datasets and is able to synchronise with the updates of those input datasets.

f) An active module may be interrupted by the user. Then it will process the specified user input and optionally wait for additional user input.

Subsequently we will use special symbols in the figures to distinguish the different types of input and output, as shown in Figure 4.4.3.

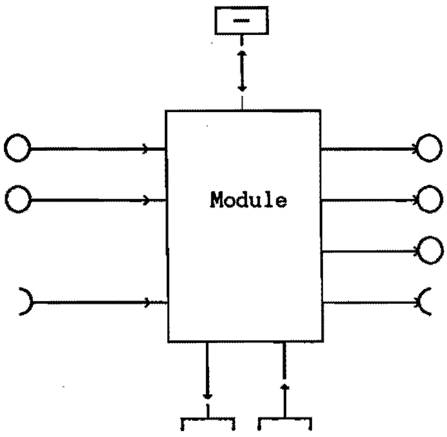


Figure 4.4.3:
Different types of module input and output.

⊟→ , →⊟	parameter input and output
]→ , →[	import and export of data
○→ , →○	dataset input and output
)→ , →(	direct input and output

The Monitor

The Monitor is the central PRIMAL component that implements the central user interface and the module management.

All user input is directed to the Monitor, unless the latter has explicitly transferred input control to a module. In the first case, the user input is interpreted by the Monitor and the task specified by the user is either carried out by the Monitor directly, or by one of the modules, in which case the Monitor activates the module, which starts executing in parallel to the Monitor. The Monitor then transfers control of the keyboard and the screen to the module, so that the module may employ direct user input and direct output. After transferring control back to the Monitor, the module continues its tasks, while the Monitor prompts the user for new input. The user may now exploit the property that modules may execute concurrently, by starting additional modules.

The Monitor tasks are summarised below:

a) It controls the communication with the user.
b) It activates modules at user request and manages the active modules.
c) It passes parameters to a module and may grant it temporary access to the keyboard and the screen.
d) It handles parameter output by the modules.
e) It manages the Database.

The Database

The PRIMAL Database contains the datasets that are produced by modules (and the Monitor) and has a hierarchical structure, see Figure 4.4.4. At the root is the root folder, containing a list of all users of the PRIMAL system. Each user has his own user folder, containing a list of different session folders, that contain the datasets produced by the modules (and the Monitor).

The Monitor maintains the Database folders. The modules produce datasets and provide the Monitor with the information necessary for maintaining the Database.
There are different types of datasets, called Vector, Matrix, Text, and Monitor datasets. Each type has a fixed organisation and interpretation.
Monitor datasets are only produced and used by the Monitor. They are used for database management, logging, etc.
The other dataset types are only used by the modules. Vector and Matrix datasets store numerical output data objects of a module. Text datasets are mainly used for logging and presentation of results to the user.
Important is that datasets are self-documenting. A module using an input dataset requires no additional information from the module that produced the dataset.

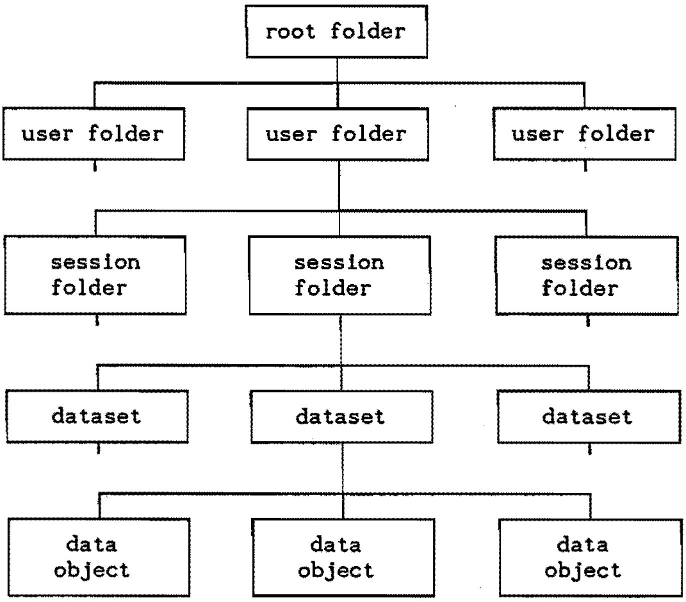


Figure 4.4.4:
Structure of the PRIMAL Database.

Dynamic datasets

When a new dataset is created, it automatically becomes a dynamic dataset, which turns into a static dataset when it is closed for updates by the module that produces it.
There are two types of dynamic datasets:

1. a dataset where new data is always appended at the end of the dataset.

2. a dataset where new data replaces existing data.

For dealing with dynamical datasets, special synchronisation mechanisms apply. See Section 4.5 for details.

4.4.3 Discussion of the design

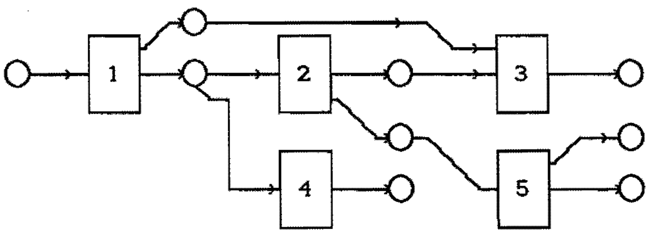
In this section we will discuss the design and the realisation of the requirements.

The Monitor activates modules and acts as the package manager, managing the Database and granting resources to the modules. It is not actively involved in the module's computations, or the data sharing and synchronisation between modules. It is not even "aware" of the function of the modules, but just implements a framework for the execution of modules (for whatever purpose). The Monitor thus acts as a special-purpose operating system. The field of applications of PRIMAL is determined by the set of modules.

4.4.3.1 Chaining of modules

Since all methods in the EM-scheme and most PRIMAL utilities, (e.g. graphical presentation, data import and export) are implemented as modules, carrying out the steps in the EM-scheme is, from a software point of view, equivalent to applying modules to the results of other modules.
Modules are independent and "unaware" of each other's existence; they only interact by sharing datasets. By taking an output dataset of one module as input to other modules, it is possible to connect modules through datasets in series and in parallel.

The ability to *chain* modules is a key property of PRIMAL.
Since the data flow is unidirectional, this results in *tree*-like structures, as shown in Figure 4.4.5.



○ = dataset

□ = module

Figure 4.4.5:
Example of chained modules.
The structure is called a tree although its branches may merge. Observe the serial connection of Modules 1,2 and 3, with merging branches as input to Module 3. Modules 2 and 4 are connected in parallel to the same input dataset.

If the datasets are static, the tree simply represents the order of the operations (as indicated by the arrows) .
If a certain dataset is dynamical, all subsequent modules run *concurrently*, *synchronised* by the dataflow.

The term "real-time" is avoided in this context. Since measured data is completely buffered in datasets, the only requirement is that the Experiment Control Module keeps in pace with the experiment. All other modules may lag behind. Since the number of active modules may change freely, the workload may also vary drastically. If the workload is too high to keep up with the experiment, modules lag behind and thus process "historical" data. If the workload drops, these modules catch up again and synchronise with the updating measurement data.

To make use of the ability of modules to synchronise with datasets and process "intermediate" results of other modules, the modules should be designed in such a way that intermediate results are meaningful and suitable for presentation or further processing.

As pointed out before, the Monitor may activate new modules at any time upon user request, by attaching them to an already existing tree. Thus, the tree is itself a dynamic structure. If a module has finished its work, it stops and its output datasets are closed for updating. This is noticed by the modules using these datasets as input. They will also stop execution after finishing their work. The tree thus "dies out" automatically from left to right.

The discussion above has concentrated on a single tree, but several independent trees of modules may be set-up in parallel.
For an illustrative example, see below.

Example 4.4-1 : On-line analysis of measurement data.

> An Experiment Control Module (ECM) (see Figure 4.4.6) generates a dataset to which new data is appended at each sample moment. A recursive data conditioning module (M1) picks up this data and generates a new dataset with the corrected data, which in its turn is picked up by a recursive identification method (M2). The current model parameter values and the measured process inputs may, likewise, be input to a recursive simulation (M3) producing a dataset with the updating model outputs and process outputs, so that the user may view the process outputs and model outputs simultaneously in real-time.
>
> As the experiment stops, the ECM stops and its output datasets become static datasets. After processing the final measurement data, the data conditioning module will stop, to be followed by the other modules. The tree "dies out" as described above. The presentation module remains active, presenting static (non-updating) pictures of the data.
>
> The user may, for instance, at any time apply a second identification method to the corrected data. This module would then start to process the already available data and the moment it catches up with the experiment, it synchronises automatically with the incoming data.

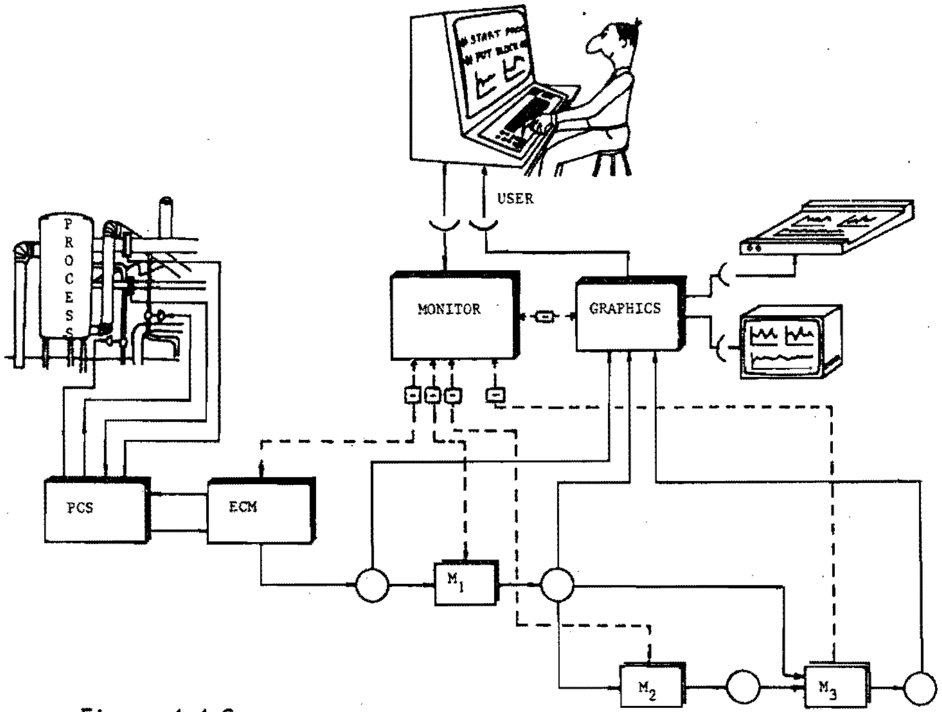


Figure 4.4.6:
Inspection of intermediate results.

The ability to access and synchronise with dynamic datasets is also important in processing data off-line.
For instance, the intermediate results of an iterative method may be written to a dataset and be inspected by the user.

4.4.3.2 The requirements

In the following, we briefly describe how the requirements listed in Sections 4.1 and 4.2 are satisfied and mention some additional properties of the design.
Requirements concerning the user interface will be treated seperately in Section 4.4.3.3. The software design requirements are discussed in Section 4.5.

a) Experiment design, experiment control and data acquisition are incorporated in the package straightforwardly as special modules. The Experiment Control Module directly interacts with the PCS to start, stop, and change experiments, and to collect and pass the measured data to an output dataset. This dataset updates at the sample moments and then makes the new data available to other modules.
Section 4.6 further describes the design of the Experiment Control Module.

b) The chaining of modules through dynamic datasets provides the facilities for real-time analyses. Parallel execution of modules allows simultaneous application of different analyses.

c) Active modules can be interrupted by the user, offering the opportunity to modify the running experiment, the presentation of data and the parameters of active analysis methods.

d) Full functionality is achieved by implementing modules for each method in the EM-scheme. The set of modules may be enlarged simply by just adding modules, without a need to change any already existing modules.

e) Since modules are mutually independent, and PRIMAL employs dataset types which have a standard structure and are self-documenting, modules are capable to read a dataset regardless of its origin. This provides the required flexibility for the user in determining his own path through the analyses.

f) The mutual independence of modules guarantees that a failing module primarily affects its own execution. Of course other modules using its output datasets are affected, but they will terminate execution normally as their data source dries up. More importantly, "upstream" error propagation is not possible. This is of vital importance to package safety. The Experiment Control Module can never be affected by errors in any analysis method using the experiment data.
Also, since the Monitor is independent of the modules, the user interface remains present and is unaffected by a failing module, see Section 4.5.

4.4.3.3 The user interface

When PRIMAL is started, the Monitor is activated and prompts the user for input. To provide flexibility, the Monitor supports a command language.

Question-and-answer or menu-driven interfaces are usually easier to learn than a command language. However, these interfaces often prove to be too rigid and too input/output intensive for experienced users. A command language shifts the inititiative to the user, requires less input/output, and is better suited to cope with incorrect user choices.

The concentration of user input and command language parsing in the Monitor ensures consistency and uniformity. The modules are not bothered with command language parsing and error recovery, which simplifies their design.

Since the Monitor is independent of the modules and is not involved in numerical computations, it ensures direct response to user input no matter the package workload. This property is important in experiment control, see Section 4.6.

To reduce the learning effort of the user, a command language has been designed with the following properties:

a) It is concise.
b) Its commands are phrased in a natural language style.
c) It obeys strict syntax rules.
d) It provides on-line help concerning commands and syntax.
e) It provides error checking and error-correction assistance.
f) It has facilities to recall and re-edit commands.
g) It is extensible by the user with new commands.

The strict syntax rules are realised by using an LL1 grammar for the language, [Wirth, 1976]. Such a language may be parsed using a left to right scanner with one-symbol look-ahead, and allows for rigorous error checking and error correction support facilities.
Each command begins with a command verb and is followed by a set of specifiers, e.g. :

```
# APPLY CORRELATOR TO FILTER.DATA
# STOP PROCESS
# CHANGE RPE (SAMPLES=400)
```

The exact syntax of the language may be formulated in Backus-Naur diagrams, cf. [Van der Linden & Renes, 1989b].

The language consists of a limited number of commands, that may be arranged in different categories:

Module management
Activating, stopping and aborting modules.
Changing the parameters of active modules.

Database management
Creating and switching folders.
Copying, moving, deleting, purging, renaming, restoring, labelling, importing and exporting datasets.
Editing datasets.
Listing folders.
Entering user messages in the log book.

Presentation of datasets
Listing datasets on screen.
Printing datasets on hard-copy devices.
Graphical presentation of datasets on screen or hard-copy devices.

Command datasets
Executing, editing, interrupting and continuing command datasets.

Informational
On-line assistance, news, debugging options.

Package control
Leaving the package.
Passing commands directly to the operating system.

The user has facilities to define new commands, composed of standard commands. These commands may include a new list of specifiers.

Parameter input to a module

The Monitor parses and executes user commands. If a command relates to a module, the Monitor passes the command to the module and suspends keyboard input and screen output; the module then gets access to the screen and keyboard until it explicitly returns the access to the Monitor.
Modules support two alternative mechanisms for parameter input.

1. Parameter list input.
   Each module has a defined set of user-adjustable parameters. The user may specify these parameters in a command directed to that module in a parameter list, which consists of parameter names and value lists:

   ```
   (parameter name = value list, parameter name = value list, ... )
   ```

   Examples:
   ```
    # APPLY RPEM TO PROCESS (SAMPLES=200,INPUT=FLOW1,OUTPUT=TEMP4)
    # PLOT P1=RPEM.DATA (XZOOM=0:100,LINETYPE=STAIRS)
   ```

   The Monitor parses the parameter list and passes its contents in a message to the module. A parameter name and its associated value list define a data object. The parameter list may thus be viewed as the argument list of a procedure call, but with indifference to the order of the arguments.

2. A cursor controlled menu with default values, questions and answers. In PRIMAL such menus are called question pages.
   A question page consists of (question) texts and answer fields. The user may jump in arbitrary order from field to field and adjust any parameter values.

These parameter input mechanisms may be combined.
In the case a user does not employ a parameter list or provides erroneous input, the module will present the question pages to guide him and request correct answers. The default answers to the questions represent reasonable standard choices of the adjustable parameters.
Experienced users may fully exploit the features of the command language and parameter lists. In this case question pages will not be presented, resulting in minimal screen output and fast module response.

The question page system has four other important features:

a) Answers are checked at input for correctness and consistency with the answers to all other questions.
b) Answers to different questions may be interdependent.
   An answer may therefore influence the correctness of the answer to another question. In the case of inconsistency the question giving rise to the conflict must be answered anew.
c) To guide the user, the "legal" answers may be displayed upon request.
d) The question pages may contain dynamical questions, i.e. questions may appear or disappear depending on answers to other questions.

The user interface of a module is activated at module start-up and when the user invokes a command for intermediate parameter changes. In this case the module will suspend its computations, process the user input and then resume execution.

Package output

The PRIMAL package prepares permanent output in the form of datasets.

The Monitor maintains the Database folders and keeps a log book for each user-session.
The log book stores and timestamps all user commands and package messages generated by the Monitor and by modules to signal errors, warnings or specific events. For example, the Experiment Control Module sends messages to the Monitor to indicate changes in the experiment, such as a change of test signal. The user himself may also enter text into the log book. With the log book and the Text datasets produced by the modules, the user can completely reconstruct the analyses and reproduce the obtained results at any later moment.

A module produces a Text dataset (in addition to the datasets with its numerical results) for the following purposes:

a) For storing the names of the input datasets and the user-adjustable parameters, usually in the same format as in the question pages. The Text dataset thus serves as the module's log book. Each time the user adjusted the parameters, the changes and the moment of change are also stored.
b) For presenting module results contributing directly to the user's learning process. If, for instance, an identification module produces a Matrix dataset containing full information about the estimated model, the Text dataset may be used to present results, e.g. in the form of the gains, time constants and relative output error(s) of the model.
c) For storing intermediate results in text format. The dataset may be used to reflect the current status of the module.

Presentation of results

The different steps in the EM-scheme generally require extensive computations involving large amounts of data, and resulting in large amounts of new data. Therefore, the package must provide powerful and flexible tools for presenting data.
PRIMAL concentrates these tools in special modules for graphical presentation, listing on the screen and printing on hard-copy devices, that allow for a high degree of user interaction to manipulate the presentations. This relieves the module programmers from this task and ensures uniformity.
The module programmer may define default presentations to show the most interesting results.

The presentation modules are capable of dealing with dynamic datasets: the picture on the screen will be automatically updated at each update of the dataset.

4.4.3.4 Learning

The PRIMAL package supports the learning process by providing:

a) Access to a variety of modules implementing different approaches to the experimental modelling problems, whose results are presented in a format allowing direct comparison.
b) The opportunity to apply several modules simultaneously to the data and inspect the intermediate results (in real-time).
c) The facilities to chain modules (in real-time).
d) An effective user interface, providing guidance and rigorous error checking/recovery.
e) Powerful interactive facilities for presenting and manipulating (real-time) data.
f) Text datasets and default graphical presentations that enhance the interpretation of the results.
g) Access to the Database, allowing retrieval of previously obtained results.

4.5 Software design strategy and implementation aspects

Top-down strategies for software projects typically consist of the following consecutive phases:
- requirements analysis
- development of design concepts
- full specification
- coding
- testing of the final product

In research projects this approach is generally not completely adequate, since the package requirements are incomplete at the project initiation. In this case an incremental software development strategy, including prototyping, is more appropriate. PRIMAL has been designed using the latter.
Making use of the experience with a first prototype PRIMAL package on a PDP-11/34 for real-time experimenting [Betlem & Rademaker, 1979], a new set of requirements was formulated, on the basis of which a specification and a radically new PRIMAL prototype have been developed in 1984. The experience with this prototype, implemented on a PDP-11/23+ under the operating system RSX-11/M, in experimental modelling of laboratory processes, led to an extended set of requirements. New functionality was added to the package without changes in its internal structure.
A next generation implementation of PRIMAL and a major design overhaul, resulting in a new ordering of functional layers, was carried out mainly in 1986, using a VAX-730 under the operating system VMS. Since then, attention has focussed on the development of

new modules for the different steps in the EM-scheme and extended graphical facilities. The core of the package has undergone minor modifications only.
In 1988 a workstation version of the package has been released for VAX-stations. The package has been applied to a variety of projects in several industries and is commercially available since 1988.

4.5.1 Software design considerations

Two important software issues in the design of the PRIMAL package were: how to meet the *safety requirements* owing to application in industrial practice, and how to ensure a high *software quality* in a large software project stretching over several years and including many relatively inexperienced programmers.

Software quality is determined by several factors that may be grouped into external factors and internal factors [Meyer, 1987].

The external factors relate to qualities whose presence or absence may be detected by the user. They include correctness (the ability to perform according to specification), robustness (the ability to deal with unspecified situations), extensibility (the ability to adapt to specification changes), compatibility with other packages, portability to other hardware, efficiency, verifiability, integrity and ease of use.

The internal factors are only perceptible to the software designers. They include adaptability of the code, readability, and comprehensibility.

In PRIMAL, two principles have been used that have large influence on the internal and external quality factors. These principles are a *clear design concept* and a *modular software architecture*.

The clear design concept is realised by the definition of only three components (the Monitor, the modules and the Database) and the rigorous use of modules. All operations on the data are implemented in the modules.
The modules are separate entities, which may be developed, tested, and used independently of each other, in fact, even outside the PRIMAL environment. To the Monitor all modules are equivalent and may be chained in any conceivable way.

The software architecture of the package is characterised by functional decomposition. In PRIMAL we distinguish clusters of basic tasks, i.e. handling command language input, direct user input/output, database management, dataset access, synchronisation with dynamical datasets, and graphical presentation. The mechanisms for carrying out these tasks are designed as separate pieces of software, which are called "software subsystems" to avoid the word "modules".

The purpose of this decomposition is to split up the design problem into smaller subproblems, whose solution may be pursued separately.

A subsystem is designed to perform a single task, with a small and well-defined user interface.
To the programmer a subsystem appears as just a set of procedures and programming conventions, called the subsystem interface. The subsystems provide encapsulation (information hiding), shielding off the implemented mechanisms and their internal data structure.
Such functional decomposition may be repeated at each level and the subsystems themselves may thus be decomposed into smaller units, using concepts like cohesion and coupling as guidelines.

The subsystems are grouped into two layers, cf. Figure 4.5.1. The first layer contains the interfaces to the standard subsystems. In PRIMAL, separate subsystems (tools), are developed for all interactions between a module and its environment. The module must use these tools for all input/output.
The second layer contains the basic mechanisms, such as for synchronisation, communication primitives and device input/output primitives. They match PRIMAL to the specific operating system and hardware devices. This layer contains the Virtual Operating System.

Layer 1 contains the following functions (tools):

- Scanning and parsing command language input
- Parameter list decoding and question page input
- Dataset access
- Message interchange between the Monitor and the modules
- Synchronisation with dynamical datasets
- Database management
- Module management
- Direct user input/output
- Model manipulation and matrix operations

Layer 2 contains the following functions:

- Synchronisation primitives
- Device input/output
- Basic dataset operations
- Message interchange
- Basic module management
- Time functions
- Symbolic name manipulation
- Data conversion and bit functions
- Exception handling

Just to give an impression of the size of the PRIMAL system, it presently consists of about 40 modules, and contains about 150000 lines of FORTRAN, of which approximately 20 % serves self-documentation. The subsystem interfaces contain about 300 procedures.

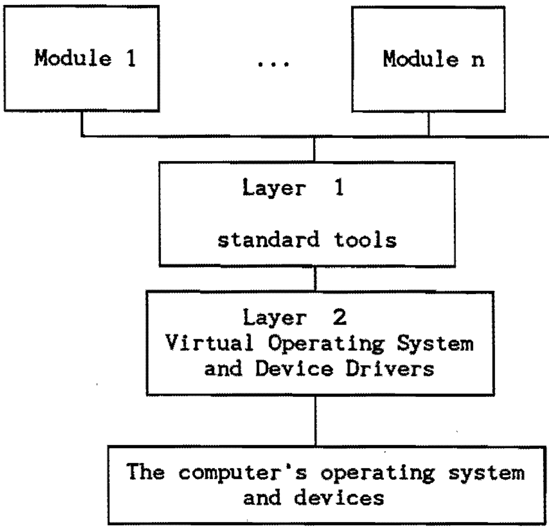


Figure 4.5.1:
The software structure of the Monitor and the PRIMAL modules.

Remarks:

a) The standard tools reduce the complexity of designing a module, and its designer needs not have a detailed understanding of the subsystem's implementation.

   The concentration of the command language interface and Database management in the Monitor, and the dataset presentation in special presentation modules, further reduces module complexity. Not having to design user i/o relieves the designer from a major programming effort, enabling him to focus on the module's primary tasks, thus promoting fast prototyping.

b) The software structure enables incremental development of the subsystems, which may be designed and updated independently of each other.

c) The use of the standard subsystems in all modules ensures a uniform interface to the user.

d) Changes and enhancements of the tools apply to all modules directly, with no adverse effects on the code that defines the modules.

4.5.2 Discussion of the software requirements

Safety

Several measures contribute to the safety of the package:

a) Most importantly, PRIMAL is split up into modules, which are implemented as independent programs, cf. Section 4.5.1. The operating system prevents, often by means of special memory protection hardware, that a failing module affects other modules. Thus, if an analysis module fails during an experiment, the data acquisition is *not* disturbed.

b) The package includes special measures to secure the consistency of the central user interface. If a module fails, the Monitor is not affected and the user maintains control over the remaining active modules.

c) The standard subsystems limit the propagation of errors by employing an error-insensitive message protocol, rigorous error checking of all input, and internal consistency checks.

d) The Database is disk-based and secures consistency even if the Monitor (or computer) fails. Modules have no direct access to the Database organisation and therefore cannot disrupt it.

e) The execution of experiments is guarded by special safety measures, see Section 4.6.

Because modules only exchange data through datasets and the dataflow is unidirectional, a failing module can only affect its output datasets and therefore indirectly the modules connected to them. If a module fails, its output datasets are usually no longer updated. In this case the downstream modules stay waiting for new data to arrive. By issuing stop commands to the waiting modules the tree of modules stops in a regular way. Upstream error propagation is not possible.

Remarks:

- If a module has claimed an input dataset for exclusive access, see Section 4.5.1, it may fail to release the dataset, effectively blocking other modules from accessing the dataset. Vital modules therefore time-out on a claim operation.

- If a module detects an error, it reports the error to the user and to the Monitor and stops in an orderly manner. The Monitor will mark the datasets produced by the erroneous module and enter the message in the log book.

Real-time aspects

The modules of PRIMAL are all designed to deal with dynamic datasets. Static datasets may be treated just as dynamic datasets, the difference being hidden at a low level in the standard tools. To the modules the real-time (on-line) or off-line use makes no difference.
The implementation of the synchronisation facilities is discussed in Section 4.5.3. The real-time aspects of performing experiments are discussed in Section 4.6.

Software management

The modular software design makes it possible to divide the design project into three phases:

1. The design of the core system, consisting of the standard tools.
2. Design of the Monitor and the special presentation modules.
3. The design of the individual modules.

The construction of PRIMAL started with phases 1 and 2, and was realised by a small group of programmers. Phase 3 is open ended and involves many programmers, usually each responsible for one or a few specific modules.
The programmer must use the standard tools and he must conform to a software documentation and lay-out standard.

Each module and basic mechanism is labelled with a release number that uniquely identifies its implementation. This number is tested at start-up of each module to check compatibility with the current PRIMAL release.

Special management modules and Monitor datasets define the "site" (the hardware on which the package is implemented) and the set of modules belonging to the package. The Monitor reads in the datasets at start-up. Changes can be made without affecting the source code.

Prototyping

The most important requirements for rapid prototyping have already been discussed. Modules are stand-alone programs, that may be developed, tested, and executed outside PRIMAL, independently of the Monitor or other modules.

Extensibility

The number of modules constituting the package may be freely extended at any time and does not require adaptation of the source code.

An important feature is the ability to install multiple modules for different implementations of the same task, such as experiment control for different PCS systems, instead of designing one complex module for all PCS systems.

Open system

PRIMAL may be extended with modules that use external input and output to communicate with other software packages. Special import/export modules may be added to PRIMAL, which translate the files used by other software packages to PRIMAL datasets and vice versa.
A more powerful way to couple with another package is to treat it as a PRIMAL module, activating it by the Monitor and providing it with the facilities to read/write PRIMAL datasets. In this way the coupling can be made fully transparent.

Portability

To enable the package to be ported to other computer systems, a wide-spread and standard programming language (FORTRAN-77) was selected. All system and device dependencies were concentrated in a single software layer, consisting of sets of basic functions, which must be modified for implementation under a new operating system.

Multi-user support

Several users may run PRIMAL simultaneously. In the Database each user has his own set of folders. He may, however, also access the datasets of other users, even though the Database may be spread over different (remote) computers.

4.5.3 Implementation

Implementation of modules

A module is implemented as an independent program, that is activated by the Monitor and is executed concurrently with the other modules and the Monitor.
A normal (non-resident) module, [Van der Linden & Renes, 1989e], stops execution automatically after finishing its computations.

As a PRIMAL module is activated it generally goes through a series of standard steps. Figure 4.5.2 lists the general structure.

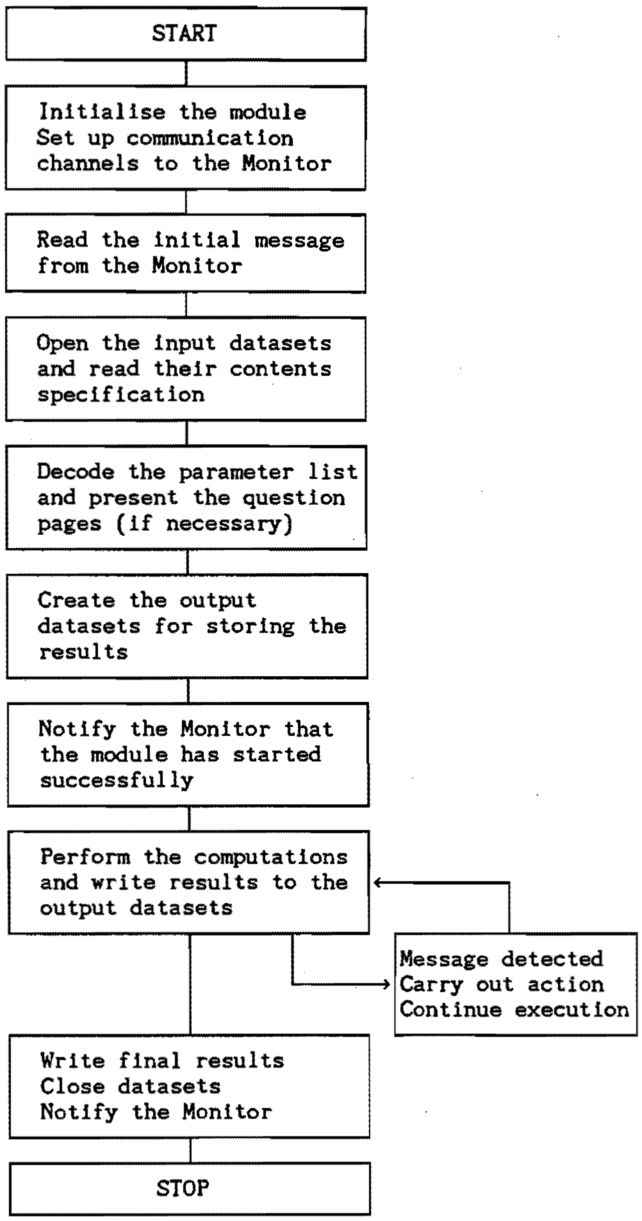


Figure 4.5.2
General structure of a PRIMAL module.

After starting, setting-up the communication channels to the Monitor, and processing the initial message sent by the Monitor, the module gains access to the keyboard and the screen, and the Monitor suspends execution. The user now interacts directly with the module. This phase is mainly used for question page input. After this phase the output datasets are created by the module for the storage of its results. The module then sends a message to the Monitor that it has started successfully and returns access to the keyboard and the screen to the Monitor. It continues its execution in parallel to the Monitor. As it has finished its computations, it closes the output datasets for write access and notifies the Monitor that it stops.

Communication between Monitor and modules

The Monitor and a module communicate by exchanging messages.
Since they are implemented as independent programs, this form of communication must take place between concurrently executing programs and it has been realised by a mailbox mechanism in combination with events and interrupts.
The Monitor uses mailboxes to receive messages from the modules. When a module is activated by the Monitor, it sets up a mailbox to receive messages from the Monitor.

Synchronisation

The package uses the following synchronisation primitives:

Events

In PRIMAL, an "event" indicates a certain occurrence. Modules may wait for a specific event, to be released from their wait state when the event is set.

If a module writes to a dynamical dataset, a synchronisation mechanism is needed to signal the reading modules that new data has been written to the dataset. Therefore, each dynamical dataset is associated with an event.
When the reading modules try to access new data that is not yet available in the dataset, they enter a wait state. When the writing module has written a set of new data to the dataset, it sets the event. The reading modules waiting for the event are released from their wait state and may then attempt to access the data. The event is reset by the *writing* module, which therefore needs not be aware of the existence of reading modules.

Semaphores

Semaphores manage the access to common resources. They are used to protect datasets from simultaneous read/write access.

When a module requests exclusive access to a common resource it initiates a wait operation on a semaphore. If it gets access it "claims" the semaphore and other modules requesting access are put in a wait state. If the module no longer needs access to the resource it releases "unclaims" the semaphore and sets an event. Waiting modules may now gain access.

Signals

Signals are used for interrupting a module. They are set by the Monitor to indicate the presence of a message or by the module itself (in an interrupt handler) to indicate user input. If a signal is raised, the module will cancel waiting for semaphores or events. The module acknowledges the signal by resetting it. Important is that a signal cancels wait states. The user may therefore interrupt an active module and get its immediate response.

All three synchronisation primitives are implemented with the help of global event flags in VAX/VMS.

Datasets

A dataset where new data is always appended to the end of the dataset is called an *access mode* 2 dataset. Modules reading the dataset may freely read all data that is already available.
The writing module indicates the availability of new data by setting an event. Access mode 2 datasets are used, for instance, for storing experiment data.
Since handshake mechanisms are avoided, a fatal error or freeze of a reading module does not affect the writing module, preventing upstream effects and possible deadlocks. Furthermore, writing modules are not blocked or delayed by reading modules, which is important to experiment control.
The mechanism described above allows a reading module to miss updates of a dynamical dataset, since it may not yet be waiting for the event. This has no influence on the correctness, since all data is stored.

A dataset where new data replaces existing data is called an *access mode* 3 dataset. Here the writing and reading modules must claim the dataset for exclusive access (using semaphores). When a dataset is claimed, the other modules requesting access enter a wait-state until the dataset is released. An update of the dataset is indicated by an event set by the writing module.
Access mode 3 datasets are used for storing intermediate results of modules, such as correlation functions and intermediate models. If a module gets access to the dataset, it will automatically get the latest update.

The wait states mentioned above are interrupted (by signals) if the Monitor issues a message to the module. The actions specified in the message must be carried out before returning to the wait-state. This procedure ensures the immediate response of the module to user commands.

Datasets are implemented as shared datafiles residing on a disk. They are self-documented and consist of three sections (with some exceptions for Monitor datasets). The first section is called the base section and contains information about the dataset size and type. The second section is type-dependent. It contains the names and characteristics of the data objects stored in the dataset and default ways of presentation. The third section is the data section, containing the stored data objects.

The Database

The Database consists of sets of files. Its folder directories are implemented as Monitor datasets, created and maintained by the Monitor.

The Graphics

Because the user's learning process depends to a large degree on the graphical presentation facilities, special attention has been devoted to their development.

A module does not present its results itself, but the Vector datasets it produces may be presented by the PRIMAL Graphics. The datasets may contain default presentation settings. The Graphics reside in a special-purpose module, which provides the facilities to:

- Present up to four independently dynamically updating presentations (pictures) simultaneously. Thus, during an experiment, the user may view the updating measurement data and the updating results of the analyses simultaneously, in real-time.
- Create several different updating presentations of one dataset.
- Make a selection from among the default presentation settings associated with each dataset.
- Manipulate the pictures using the command language and/or graphical input, supporting a large set of operations, e.g. for selecting presented curves, zooming, line types, accuracy bands, scaling, etc.
- Present the pictures on a variety of graphical devices.

The graphical subsystem consists of three layers, implementing a effective set of high-level plot operations, a device selection layer, and a collection of graphical device drivers, [Van Lanen, 1988].

4.6 Experiment design and data acquisition

Starting-point for the design of the process interface is the computer configuration introduced in Section 3.3.2, (Figure 3.3.1). The different tasks that may be discerned in experiment design and experiment control are divided over the Analysis Computer (PRIMAL computer) and the PCS.
The PRIMAL role is based on a standard set of messages that must be exchanged with the PCS and is therefore to a large degree PCS-independent. This concept makes it possible to couple PRIMAL to a large diversity of process control systems, at the cost of developing an application-specific PCS part of the interface for each new type of PCS to which a coupling must be realised.

In PRIMAL, all activities concerning experiment design and experiment control are handled by a set of modules:

1. The Interface Definition Module (IDM)
2. The Experiment Definition Module (EDM)
3. The Experiment Control Module (ECM)
4. The Process Interface Module (PIM)

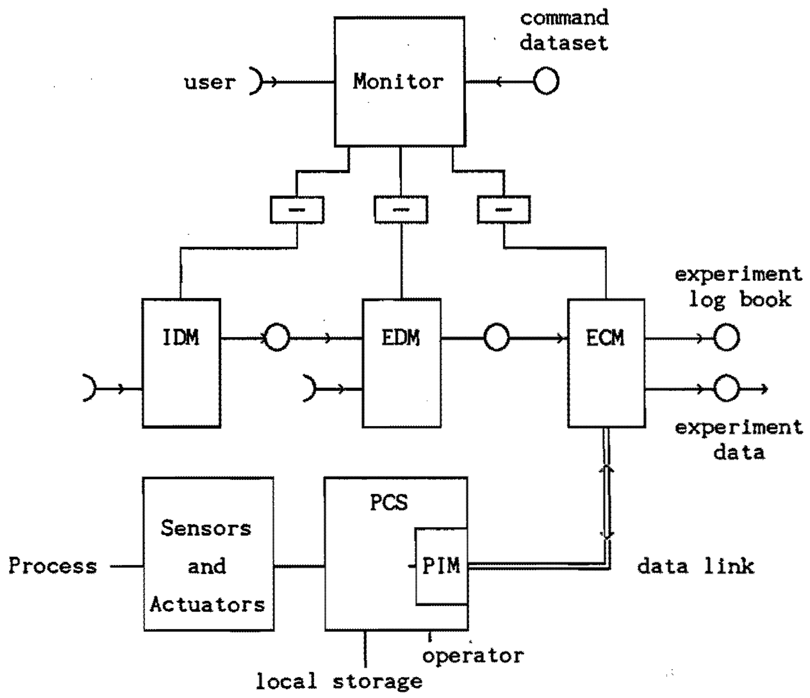


Figure 4.6.1:
Overview of modules involved in experiment control.
The ECM loads the experiment definition and passes it to the PIM. The experiment data are stored locally in the PCS and/or in a dataset by the ECM.

The Process Interface Module runs in the PCS. The other modules run in the Analysis Computer as parts of PRIMAL.

The Process Interface Module

The Process Interface Module in the PCS has no direct link to the Monitor. It communicates only with the Experiment Control Module (ECM) and it is not implemented as a PRIMAL module.
The main task of the PIM is to start/stop the data collection (of the sampled process inputs/outputs) and transfer the data to the ECM, which makes it available to all other PRIMAL modules.

The activities of the PIM depend on the selected options. The module should be capable to operate in combination with PRIMAL as well as independently, supporting design and control of experiments by PRIMAL, as well as local operator control. At any time the operator may interrupt an experiment and override PRIMAL's actions.

In short, the PIM performs the following tasks:

a) It communicates with the ECM on the Analysis Computer.
b) It initialises its actions by loading the "experiment definition", which is passed to the PCS by the ECM, unless it has already been stored locally.
c) It schedules experiment control commands to start, stop and change experiments, sent by the ECM, or entered locally (e.g. by the operator). Local commands override remote commands.
d) It collects the values of the process inputs and outputs at each sample moment and passes these to the ECM and/or stores them locally.
e) Optionally, it generates and introduces test signals into the process inputs.
f) During the experiments, it checks whether all process inputs and outputs stay within their legal ranges, refusing to introduce input signals in the case of conflicts.
g) It logs all actions and relevant experimental conditions in a local log book and/or sends messages to the Analysis computer for remote logging.
h) Optionally, it implements controllers, the controller parameters being passed to it by PRIMAL or entered locally.

A prototype PIM has been developed, and specific implementations were carried out on the laboratory PCS of the Systems and Control Group at Eindhoven University of Technology, on a Foxboro Microspec [Renes & Van der Linden, 1987], and, in cooperation with Philips, on a special-purpose PCS for process identification.

The Interface Definition Module and the Experiment Definition Module

The parameters defining the experimental set-up and the experiments are stored in an interface definition dataset and an experiment definition dataset. The interface definition for a specific PCS-process combination is usually made once, whereas the experiment definition may change in each experiment. Therefore, a separation of tasks is made for reasons of safety and convenience.

The IDM produces the interface definition that is used by the EDM in producing the experiment definition. The interface definition contains information such as: the available PIM options, the maximum number of process inputs and outputs, the type and range of the ADC's and DAC's and the range of addresses that identify the process inputs and process outputs.

The experiment definition contains information such as: the sampling interval, default experiment length, the selected PIM options, the selected inputs and outputs, their default, minimum, maximum and warning levels, prefiltering options, etc.

The Experiment Control Module

The Experiment Control Module has two parts. The kernel is constructed as a PCS-independent PRIMAL module, using the standard tools for communicating with the user and accessing the datasets. The other part is PCS-dependent and implements the specific requirements of the communication protocol with the PIM.
Any new PCS may be coupled to PRIMAL simply by adding an appropriate ECM to the package. Each ECM is assigned a unique name, by which it is known to PRIMAL.

In short, the ECM performs the following tasks:

a) It loads an experiment definition dataset.
b) It sets up the communication channel to the PIM.
c) It sends the experiment definition parameters to the PIM.
d) It waits for commands from the Monitor and from the PIM.
e) When a request arrives to start an experiment, it asks the PIM to send process data and stores these data in its output dataset(s), automatically making it available to other modules for analyses and/or presentation in real-time plots.
f) It instantly responds to user commands to adapt or stop the experiment.
g) It stores all status information, messages from the PIM and all user commands in the experiment log book (a Text dataset), ready to be accessed by the user at any time.
h) It remains resident after the experiment is stopped, ready to be re-activated and to repeat steps a) - h).

Safety

Concerning safety, the set-up described above has the following properties:

- Only a relatively simple program (PIM) must be implemented on the PCS to couple PRIMAL to the process.
- All PRIMAL interactions with the process pass through this module, which rigorously checks whether all actions remain within the limits of the experiment definition.
- The operator may overrule remote commands.
- Failing modules in the Analysis Computer do not affect the PCS.

The two-computer concept and the built-in flexibility of the PIM and the ECM to enable/disable certain parts of experiment control are usually sufficient to comply with the safety requirements encountered in practice.

Real-time aspects

The PRIMAL package is designed to operate correctly, no matter the number of active modules and the computational workload. As long as the ECM, running at high priority, can keep up with the PIM, the analyses may lag behind, but will not conflict with the real-time demand.
By providing sufficient buffering of data sent over the data link, the real-time demands for the ECM may be relaxed; it must keep up with the average data transfer rate, but may lag behind at peak activities of the Analysis Computer. Of course, this does not hold if the ECM is used to generate test signals or to implement a controller.
To prevent the user from blocking ECM execution, the module does not allow direct user input. User interaction is only possible through commands handled by the Monitor.

Experiment control

The experiment definition dataset does not include experiment control actions, such as starting, stopping and introducing test signals. These actions are carried out interactively by the user by commands such as:

```
    # LOAD PROCESS WITH EXPERIMENT
    # START PROCESS
    # PUT STEP ON PROCESS.FLOW1 (AMPLITUDE=40)
```

See Section 4.7 for further examples.

Experiment control commands may be programmed in command datasets and may include time specifications. In this way, experiments can be programmed beforehand, to be executed automatically later.
The Monitor parses the commands and passes them to the ECM, which in turn orders the PIM to start, stop, or adjust the experiment. If the

commands indicate that an adjustment must be made at some future time, it is entered in the scheduling table of the PIM.

For experiments with high sampling rates, the experiment duration is usually short and interactive analysis and experiment adjustment may not be feasible. But then, PRIMAL may be used to switch back and forth rapidly between experimenting and analysing the data.

The experiment definition dataset, and the session and experiment log books allow the user to reconstruct the experimental conditions at any later time. The log books store all commands issued by the user and all messages sent by the PIM. Adjustments to the experiment are confirmed by the PIM.

Control system implementation

The design of PRIMAL allows a gradual implementation of new control systems. In a first step, the control system may be simulated in PRIMAL using the best process model found. In a second step, the control system outputs are computed by the Analysis Computer and passed to the PIM to be imposed on the process inputs. In a third step, the control system may be implemented in the PCS.

4.7 A sample session with PRIMAL

To demonstrate the PRIMAL package, this section discusses a sample session in the style of the user manual, [Van der Linden & Renes, 1989a].

PRIMAL sessions

As the user enters PRIMAL, he is asked for a user name and a session name. The package uses this information to create a session folder in the database of the user. All datasets produced in this session are by default stored in the session folder.

The full and unique specification of a dataset is:

[<*user name*>.<*session name*>]<*dataset name*>.<*extension*>;<*version*>

```
e.g.  [JACK.ETHENE]PROCESS.DATA;2
      [JACK.TEST1]FFT.DATA;3
```

At a given moment the directory of the session folder may look like:

```
Directory of Folder [JACK.HYDRO]

30MAR90 10:17 CORRELAT DATA       1      Cross correlations for inputs Q and QS
30MAR90 10:17 CORRELAT TEXT       1      Cross correlations for inputs Q and QS
30MAR90 10:11 EXPEDIT  IFD        1      Thermo Hydraulic process interface def
30MAR90 10:14 HYDRO    EXP        1      Definition PRBNS experiment 1
30MAR90 10:10 HYDRO    LOG        1  [L] Log-book of session[JACK.HYDRO]
30MAR90 10:39 MARKOV   DATA       1      Impulse responses Q,QS -> T3,T4
30MAR90 10:39 MARKOV   MODEL      1      Impulse responses Q,QS -> T3,T4
30MAR90 10:39 MARKOV   RESPONSE   1      Impulse responses Q,QS -> T3,T4
30MAR90 10:39 MARKOV   TEXT       1      Impulse responses Q,QS -> T3,T4
30MAR90 10:39 MTEST    CORR       1      Model validation of the RPEM;1
30MAR90 10:39 MTEST    DATA       1      Model validation of the RPEM;1
30MAR90 10:39 MTEST    TEXT       1      Model validation of the RPEM;1
30MAR90 10:17 PREFILTE DATA       1      Trend corrected process data
30MAR90 10:17 PREFILTE TEXT       1      Trend corrected process data
30MAR90 10:16 PROCESS  DATA       1      PRBNS experiment 1
30MAR90 10:16 PROCESS  TEXT       1      PRBNS experiment 1
30MAR90 10:17 RPE      DATA       1      SISO estimate Q -> T2 order 2
30MAR90 10:17 RPE      MODEL      1      SISO estimate Q -> T2 order 2
30MAR90 10:17 RPE      TEXT       1      SISO estimate Q -> T2 order 2
30MAR90 10:37 RPEM     DATA       1      Output error Q,QS -> T3,T4  order 3
30MAR90 10:37 RPEM     MODEL      1      Output error Q,QS -> T3,T4  order 3
30MAR90 10:37 RPEM     TEXT       1      Output error Q,QS -> T3,T4  order 3
```

Figure 4.7.1:
Example of a session folder directory.
The datasets in this folder are characterised by their name, a user-added label and their time and date of creation. Furthermore, the session folder includes a log book.

The user may manipulate (e.g. copy, move, rename, delete) the datasets with appropriate Monitor commands.

Examples:

```
# COPY [JACK.HYDRO]PROCESS.DATA TO [JOHN.TEST1]HYDRO1.DATA
# DELETE FFT.DATA;1
```

Performing experiments

Suppose the Experiment Control Module has been named PROCESS.
After the interface definition has been made, the user generates an experiment definition with:

```
# EDIT HYDRO.EXP
```

The Monitor will start up the Experiment Definition Module which presents a set of question pages to the user, for defining the experiment parameters, process inputs and outputs and their properties, [Van der Linden & Renes, 1989d].

The ECM is loaded with the command:

```
# LOAD PROCESS WITH HYDRO.EXP
```

The ECM sets up the communication channels to the PIM, loads the experiment definition from the EDM and passes the appropriate information to the PIM. Furthermore, the module creates a dataset for storing the data generated in the experiment, and a log book to store the experimental conditions.
The package responds with:

```
   Created (unit 1): [JACK.HYDRO]PROCESS.TEXT;1
   Created (unit 2): [JACK.HYDRO]PROCESS.DATA;1
   Module PROCESS:1 loaded
```

The user may now start an experiment with:

```
   # START PROCESS
```

Sampling will now start and the measured data will be stored in the dataset PROCESS.DATA by the ECM.
The data from the process can be accessed immediately. For example, to monitor the experiment, the data may be displayed in a real-time plot with the command:

```
   # PLOT P1 = PROCESS.DATA IN WINDOW C1
```

The user may modify the experiment interactively, for instance by specifying test signals:

```
   # PUT BNOISE ON PROCESS.PS:10 (AMPL=0.2,LAMBDA=5)
```

which informs the Experiment Control Module PROCESS to initiate a binary signal with amplitude 0.2 and a minimal clock period of 5 samples in the input signal with name PS, beginning at sample 10.

On-line analyses

While the experiment is active, the user may directly apply analysis methods to the data. For instance:

```
   # APPLY PREFILTER TO PROCESS.DATA
```

starts up the recursive data conditioning module PREFILTER to separate the trend from the raw measurement data.
With:

```
   # APPLY CORRELATOR TO PREFILTER.DATA
```

correlation analysis is applied to the output data of PREFILTER, and with:

```
   # PLOT P2=CORRELATOR.DATA IN WINDOW D2
```

the updating correlation functions are displayed next to the updating measurement data. The user may proceed immediately, for instance, by applying a recursive prediction error method to the data:

```
# APPLY RPE TO PREFILTER.DATA
```

A default presentation of the corrected process output, the output predicted by the model, and the prediction error may be shown on the graphical screen with:

```
# PLOT P3=RPE.DATA (SET=OUTPUT) IN WINDOW D4
```

The modules PROCESS, PREFILTER, CORRELATOR, RPE and the Graphical module are now simultaneously active, synchronised by the data flow. The graphical screen presents plots as shown in Figure 4.7.2.

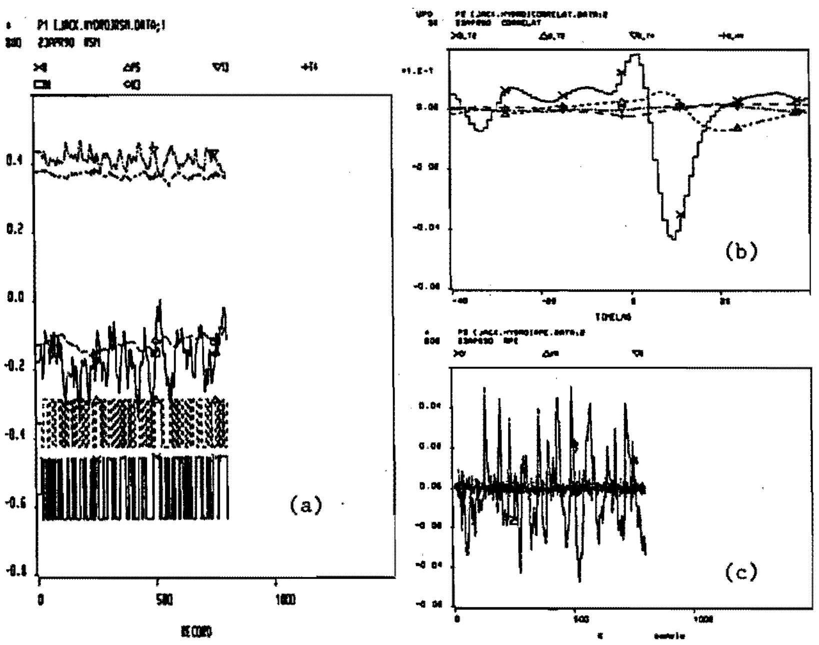


Figure 4.7.2:
Copy of the graphical screen at some moment during on-line analysis of the experiment data.
Updating pictures are present of the process data (a), estimated correlation functions (b), the process output, the model output and prediction error of a recursive identification method (c).

4.8 Comparison to existing software packages

After this outline of the design and functionality of PRIMAL, we summarise its most important features here and compare it to other software packages.

PRIMAL provides:

- Support for all steps in our Experimental Modelling scheme.
- A variety of modules for the different steps in the EM-scheme.
- Analyses in real-time.
- The ability to perform several analyses in parallel and/or in synchronised structures.
- Real-time graphical presentation of multiple datasets.
- Inspection of intermediate results.
- Command input, question page input and graphical input.
- Interaction with and adaptation of active methods.
- Full database support.
- Self-documented datasets.
- Automated log book support.
- Results presented in a form suited to less experienced users.
- Support for multiple users.
- An environment that can be extended freely with new modules.

The combination of real-time capabilities and an extensive set of modules for data conditioning, identification and control system design, makes the PRIMAL package unique in providing the facilities for bridging the gap between control theory and industrial practice. Among other software packages, it fills a gap between, on the one hand, real-time packages for experimenting and on the other hand, packages for off-line data analysis and CACSD (Computer Assisted Control Systems Design).

To the first category belong PCS-type packages, (e.g. The Fix, Genesis, ..) and analysis packages (e.g. ASYST). They provide facilities for experiment control and signal analyses, but usually very limited support for identification and control design.

To the second category belong packages such as CTRL-C, MATRIX-X, PRO-MATLAB (belonging to the MATLAB family), KEDDC and LUND. See Cellier & Rimvall [1986] for an overview.
These packages are mainly intended for off-line use, but a few, such as KEDDC and MATRIX-X, offer some real-time capabilities, which, however, are not fully integrated into the package.

Of the popular MATLAB family, CTRL-C and MATRIX-X aim mainly at simulation and control, and offer only limited support for identification. Closest to PRIMAL is PRO-MATLAB, because its identification and control toolboxes provide an extensive set of tools for modelling and control.

The main differences between PRO-MATLAB and PRIMAL concerning experimental modelling, are summarised below.

- The MATLAB user interface is essentially a language for matrix manipulation, including variables and flow control, that is more powerful and offers more flexibility than the PRIMAL command language. It is, however, less suited to inexperienced users.
- By supporting the development of programs written in MATLAB language, MATLAB provides a suitable environment for fast prototyping of programs for numerical analyses.
- MATLAB supports a large set of tools for signal analysis, identification and control system design.
  Its facilities for multivariable MFD's are as yet not well developed.
- The package has an open architecture (MEX-files).
- It has no real-time capabilities.
- Experiment design/control facilities are not available.
- Analyses cannot be carried out in parallel.
- Intermediate results cannot be inspected.
- The workspace consists of a loose collection of matrices. Data can be structured to some extent within a matrix only.
- No structured database is available, so that a convenient storage of results and reconstruction of the analyses is not supported.
- No facilities are available for (automatically) generating reports of the results of the different methods.
- Data structures are insufficiently standardised, especially among the different toolboxes.

Concluding, I think that, in their current form, PRO-MATLAB is an excellent tool for exploring and developing methods, whereas PRIMAL, owing to its interface to the process, real-time analysis support, database and reporting facilities, is much stronger in practical applications.

CHAPTER 5 IDENTIFICATION METHODS IN PRIMAL

5.1 Introduction

Following the Experimental Modelling strategy presented in Chapter 3, the approach to identification is to provide the user with a variety of methods, based on different types of models and estimation methods, and let the user choose from among these methods, probably applying several of them in parallel and comparing their results.

In this chapter, we discuss the identification methods presently available in PRIMAL. We consider the following classes of methods:

a) (One-step ahead) prediction error methods, in their recursive, direct, and iterative variants.
b) As a special case of a), the estimation of a (high order) FIR model, followed by a realisation step.
c) Instrumental variable methods.
d) Effective new variants, based on a Monte-Carlo approach.

5.2 Prediction Error Methods

The Prediction Error Methods (PEM's) form an important class of identification methods, that is popular for several reasons:

a) PEM's are well established from a statistical point of view. In the case of a quadratic prediction error criterion they can be derived directly from the Maximum Likelihood method, assuming Gaussian distributed noise.
b) Using special choices of the general model parametrisation (2.6.4) and discerning gradient type and Gauss-Newton type optimisation techniques, Ljung [1983] has shown that many "classical" identification schemes, such as Generalised Least Squares, Approximate Maximum Likelihood, etc., are special cases of a general PEM.
c) They may be implemented in an iterative or recursive fashion, and perform reportedly well.
d) Without modification they may be used in closed loop experiments.
e) The PEM can be extended to include data prefiltering and/or other norms of the prediction error.

For the above reasons several PRIMAL modules were based on the PEM approach. In the sequel we will briefly summarise the basic iterative and recursive implementations. The general predictor model for the SISO case [Ljung, 1983] is extended here to suit the MISO case.

The parameter estimate is defined as the minimising argument of the criterion function:

$$V_N(\theta,Z^N) = \frac{1}{2N}\sum_{t=1}^{N} \epsilon^T(t,\theta)\Lambda^{-1}\epsilon(t,\theta) \qquad (5.2.1)$$

$$\hat{\theta}_N = \arg\min_{\theta} V_N(\theta,Z^N) \qquad (5.2.2)$$

For notational convenience, the subscript N and the argument Z^N are dropped in the sequel. Generally, $V(\theta)$ is a non-linear function of the parameters and its minimum cannot be computed analytically. Several numerical optimisation techniques have been developed for this non-linear least squares problem, see [Gill & Murray, 1981; Dennis & Schnabel, 1986]. We will briefly discuss the gradient and Gauss-Newton techniques.

At θ we approximate $V(\theta+\Delta\theta)$ by its second order Taylor expansion, neglecting higher order terms.

$$V(\theta+\Delta\theta) = V(\theta) + g^T(\theta)\Delta\theta \;+ \tfrac{1}{2}\Delta\theta^T C(\theta)\Delta\theta \qquad (5.2.3)$$

with $g(\theta)$ the gradient, and $C(\theta)$ the Hessian of $V(\theta)$ with respect to θ. A strong local minimum θ_o of the criterion function satisfies:

$$\begin{aligned} g(\theta_o) &= 0 \\ C(\theta_o) &\ \text{positive definite.} \end{aligned} \qquad (5.2.4)$$

Defining $\psi(t,\theta)$ as the gradient of the prediction error with respect to θ:

$$\psi(t,\theta) := -\frac{\partial\epsilon^T}{\partial\theta}(t,\theta) \qquad (5.2.5)$$

we get:

$$g(\theta) = V'(\theta) = \frac{-1}{N}\sum_{t=1}^{N} \psi(t,\theta)\Lambda^{-1}\epsilon(t,\theta) \qquad (5.2.6)$$

$$C(\theta) = V''(\theta) = \frac{1}{N}\sum_{t=1}^{N} \psi(t,\theta)\Lambda^{-1}\psi^T(t,\theta) \;-\; S \qquad (5.2.7)$$

$$\text{where } [S]_{nm} = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{q}\sum_{j=1}^{q} \frac{\partial}{\partial\theta_n}(\psi_{mi}(t,\theta))\Lambda^{-1}_{ij}\epsilon_j(t,\theta)$$

Iterative optimisation techniques

The parameter estimate θ^{i} in the i'th iteration may be updated in the steepest descent direction, with step-size $\alpha_i > 0$ using:

$$\theta^{i+1} = \theta^{i} - \alpha_i g(\theta^{i}) \tag{5.2.8}$$

or, using second order derivatives, with a Newton technique:

$$\theta^{i+1} = \theta^{i} - [G(\theta^{i})]^{-1} g(\theta^{i}) \tag{5.2.9}$$

The Newton technique requires knowledge of the Hessian (5.2.7), which consists of a positive semi-definite first term and a remainder S. Neglecting the remainder results in a Gauss-Newton scheme.

$$\theta^{i+1} = \theta^{i} + \alpha_i [\sum_{t=1}^{N} \psi(t,\theta^{i})\Lambda^{-1}\psi^{T}(t,\theta^{i})]^{-1} \sum_{t=1}^{N} \psi(t,\theta^{i})\Lambda^{-1}\varepsilon(t,\theta^{i}) \tag{5.2.10}$$

A step size α_i is included in this scheme and may be computed using a line minimisation method.
The advantage of a Gauss-Newton scheme over a Newton scheme is that it only requires the first derivatives and that the positive definite approximation of the Hessian guarantees a descent direction. However, the Gauss-Newton scheme is less suited for large residual problems and problems with a near to singular approximate Hessian. In the case of singular approximate Hessian the Levenberg-Marquardt technique may be employed to ensure a regular Hessian. For large residual problems, a modified Newton technique may be employed.

Recursive optimisation techniques

For the derivation of the recursive Gauss-Newton technique the criterion function is extended to include a forgetting factor λ.

$$V_t(\theta) = \frac{1}{2} \sum_{s=1}^{t} \lambda^{t-s} \varepsilon^{T}(s,\theta)\Lambda^{-1}\varepsilon(s,\theta) \tag{5.2.11}$$

Assume that θ_{t-1} minimises $V_{t-1}(\theta)$. $V_t(\theta)$ is approximated locally at θ_{t-1} by a second order Taylor series, neglecting higher order terms:

$$V_t(\theta) = V_t(\theta_{t-1}) + g_t(\theta_{t-1})(\theta-\theta_{t-1}) + \frac{1}{2}(\theta-\theta_{t-1})^T G_t(\theta_{t-1})(\theta-\theta_{t-1}) \qquad (5.2.12)$$

The minimising argument θ_t of this quadratic expression is:

$$\theta_t = \theta_{t-1} - G_t^{-1}(\theta_{t-1})g_t(\theta_{t-1})^T \qquad (5.2.13)$$

To obtain recursive relationships $g_t(\theta)$ and $G_t(\theta)$ are expressed in terms of $g_{t-1}(\theta)$ and $G_{t-1}(\theta)$. Straightforward differentiation of (5.2.11) results in:

$$g_t(\theta) = \lambda g_{t-1}(\theta) - \epsilon^T(t,\theta)\Lambda^{-1}\psi^T(t,\theta) \qquad (5.2.14)$$

$$G_t(\theta) = \lambda G_{t-1}(\theta) + \psi^T(t,\theta)\Lambda^{-1}\psi(t,\theta) - \sum_i \sum_j \epsilon_i(t,\theta)\Lambda_{ij}^{-1}\psi_j'^T(t,\theta) \qquad (5.2.15)$$

Near the optimimum the last term of (5.2.15) will be approximately zero if $\epsilon(t,\theta)$ converges to a white noise. This part of the Hessian may therefore be neglected, leading to the Gauss-Newton algorithm:

$$\theta_t = \theta_{t-1} + [G_t(\theta_{t-1})]^{-1}\psi(t,\theta_{t-1})\Lambda^{-1}\epsilon(t,\theta_{t-1})$$

$$G_t(\theta_{t-1}) = \lambda G_{t-1}(\theta_{t-1}) + \psi(t,\theta_{t-1})\Lambda^{-1}\psi^T(t,\theta_{t-1}) \qquad (5.2.16)$$

To arrive at an algorithm suitable for on-line use:
- the signals $\epsilon(t,\theta)$, $\psi(t,\theta)$ are replaced by on-line approximations $\epsilon(t)$, $\psi(t)$.
- $G_{t-1}(\theta_{t-1})$ is approximated by $G_{t-1}(\theta_{t-2})$.
- the matrix Λ is replaced by an on-line approximation Λ_t.
- and the matrix inversion lemma is applied.

This leads to the recursive algorithm:

$$\begin{aligned}
P(t) &= \lambda^{-1}[P(t-1) - L(t)\psi^T(t)P(t-1)] \\
L(t) &= P(t-1)\psi(t)[\lambda\Lambda_t + \psi^T(t)P(t-1)\psi(t)]^{-1} \\
\Lambda_t &= (1-\gamma_t)\Lambda_{t-1} + \gamma_t\epsilon(t)\epsilon^T(t) \qquad \gamma_t = 1/t \\
\theta_t &= \theta_{t-1} + L(t)\epsilon(t) \qquad (5.2.17)
\end{aligned}$$

The predictor model

We will use the following predictor model, based on the MISO data generating model (2.6.5):

$$\hat{y}(t|t-1;\theta) = [\ 1 - A(q^{-1};\theta)D(q^{-1};\theta)C^{-1}(q^{-1};\theta)]y(t) + D(q^{-1};\theta)C^{-1}(q^{-1};\theta)\sum_{i=1}^{p} B_i(q^{-1};\theta)F_i^{-1}(q^{-1};\theta)u_i(t) \tag{5.2.18}$$

$$\varepsilon(t;\theta) = y(t) - \hat{y}(t|t-1;\theta) \tag{5.2.19}$$

The MISO, rather than the MIMO version, is selected to reduce the computational workload in on-line operation. Then, in order to model a MIMO system q independent estimations must be carried out.
Defining:

$$w_i(t) := B_i(q^{-1};\theta)F_i^{-1}(q^{-1};\theta)u_i(t) \qquad i = 1..p \tag{5.2.20}$$

$$v(t) := A(q^{-1};\theta)y(t) - \sum_{i=1}^{p} w_i(t) \tag{5.2.21}$$

the prediction error can be written as:

$$\varepsilon(t) = D(q^{-1};\theta)C^{-1}(q^{-1};\theta)v(t) \tag{5.2.22}$$

Equation (5.2.22) can be rewritten to:

$$\begin{aligned}\hat{y}(t|t-1;\theta) &= y(t)-\varepsilon(t)\\ &= [1-A(q^{-1};\theta)]y(t) + \sum_{i=1}^{p} \{\ B_i(q^{-1};\theta)u_i(t) + \\ &\quad [1-F_i(q^{-1};\theta)]w_i(t)\ \} + [1-D(q^{-1};\theta)]v(t) + \\ &\quad [C(q^{-1};\theta)-1]\varepsilon(t)\\ &:= \varphi^T(t,\theta)\theta\end{aligned} \tag{5.2.23}$$

where:

$$\theta := (a\ b_1\ ..\ b_p\ f_1\ ..\ f_p\ c\ d)^T \tag{5.2.24}$$

Each component of θ represents the row-vector of coefficients of the corresponding polynomial.

Consequently, the vector $\varphi(t,\theta)$ consists of past signal values:

$$\begin{aligned}\varphi^T(t,\theta) = (& -y(t-1) \,..\, -y(t-na)\; u_1(t) \,..\, u_1(t-nb_1) \,..\\ & u_p(t) \,..\, u_p(t-nb_p)\; -w_1(t-1;\theta) \,..\, -w_1(t-nf_1;\theta) \,..\\ & -w_p(t-1;\theta) \,..\, -w_p(t-nf_p;\theta)\; \varepsilon(t-1;\theta) \,..\, \varepsilon(t-nc;\theta)\\ & -v(t-1;\theta) \,..\, -v(t-nd;\theta)) \qquad (5.2.25)\end{aligned}$$

An expression for the gradient is obtained by differencing (5.2.23) with respect to θ.

$$\psi^T(t,\theta) = -(\frac{\partial}{\partial a}, \frac{\partial}{\partial b_1}, \dots \frac{\partial}{\partial b_p}, \frac{\partial}{\partial f_1}, \dots \frac{\partial}{\partial f_p}, \frac{\partial}{\partial c}, \frac{\partial}{\partial d})\varepsilon(t,\theta)$$

$$:= (\psi_a^T \;\; \psi_{b1}^T \,..\, \psi_{bp}^T \; \psi_{f1}^T(t) \; \psi_{fp}^T(t) \; \psi_c^T \; \psi_d^T)$$

$$C(q^{-1};\theta)\psi_{a,k}(t;\theta) = D(q^{-1};\theta)y(t-k) \qquad k=1..na$$

$$C(q^{-1};\theta)F_i(q^{-1};\theta)\psi_{bi,k}(t;\theta) = D(q^{-1};\theta)u_i(t-k) \qquad k=0..nb_i,\; i=1..p$$

$$C(q^{-1};\theta)\psi_{c,k}(t;\theta) = \varepsilon(t-k;\theta) \qquad k=1..nc$$

$$C(q^{-1};\theta)\psi_{d,k}(t;\theta) = -v(t-k;\theta) \qquad k=1..nd$$

$$C(q^{-1};\theta)F_i(q^{-1};\theta)\psi_{fi,k}(t;\theta) = -D(q^{-1};\theta)w_i(t-k;\theta) \qquad k=1..nf_i,\; i=1..p$$

(5.2.26)

Iterative methods

The equations (5.2.10), (5.2.23) and (5.2.26) form an iterative Gauss-Newton algoritm. This scheme has been implemented in the second step of the PRIMAL module MCRPEM, [Van der Linden & Renes, 1989e] taking into account the following implementation points:

- (5.2.10) is not used directly, but instead the QR-decomposition of the associated least-squares problem is used, which also serves to check the rank of the Jacobian for the purpose of regularisation.
- The step size (damping factor) α is determined with a line minimisation algorithm, based on safeguarded quadratic interpolation.
- The Gauss-Newton method has been developed for unconstrained optimisation problems. However, θ is restricted to the subset $\mathcal{D}m$, (2.3.4), which implies that all roots of the polynomials $C(z)$ and $F_i(z)$, $i=1..p$ must lie inside the unit-circle. This projection into the unit circle is implemented in the line minimisation algorithm.

Recursive methods

Recursive algorithms based on the model (5.2.18) have been implemented in the module RPEM (for the MISO case) and RPE (for the SISO case), [Van der Linden & Renes, 1989e]. The following points were taken into consideration:

- The P-matrix in (5.2.17), which may be interpreted as the parameter covariance matrix, is computed using a UDU^T decomposition, guaranteeing a strict monotonic decrease.
- $\epsilon(t,\theta)$ and $\psi(t,\theta)$ are approximated by $\epsilon(t)$ and $\psi(t)$, using the shift properties, cf. [Ljung 1983].
- The stability of the predictor is guaranteed by a projection of unstable zero's of C(z) and F(z) into the unit circle.
- In updating $\varphi(t)$, the residual is used instead of the prediction error.
- Switching from a Pseudo Linear Regression (PLR) to a Gauss-Newton scheme has been built in to circumvent possible convergence problems when the initial estimate is far from the optimum.
- To achieve accuracies comparable to the iterative version of the PEM algorithm, re-iteration of the data has been built-in. This, of course, is not suitable for real-time application.

Some notes on equation error and output error methods

Depending on the selected parametrisation in (5.2.18) we may speak of equation error type methods ($F(q^{-1};\theta) = I$) or output error type methods ($A(q^{-1};\theta) = I$).
Comparing these types, we may note the following:

- Simple ARX-type equation error models may be estimated using ordinary least squares, see Section 5.3. Therefore, they are computationally more attractive than output error models, which usually require non-linear optimisation techniques and which may suffer from local optima.
- Output error methods explicitly minimise a direct measure for the model performance in simulation.
- An important property of equation error methods is, that for low-pass processes more weight is put at high frequencies. This is demonstrated by Wahlberg & Ljung [1986], using an expression for the limiting criterion function in the frequency domain. It has been shown by Van den Hof & Janssen [1985] that in the time domain (under certain assumptions) equation error methods fit the initial set of Markov parameters. For output error methods the weight is averaged over the frequencies.
- For output error models of type (5.2.18), where the process and noise transfer function are independently parametrised, and the system operates in open loop, it can be shown that the process transfer function converges to the "true" transfer function (assuming an exact fit is possible) even if the noise filter is not parametrised appropriately.

5.3 Least Squares Methods

For the quadratic criterion function (5.2.1) and the special case that the prediction error is linear in the parameters, the (unique) minimum of $V(\theta)$ can be computed directly, using a Least Squares method. We will briefly discuss two cases.

ARX-model

In the case of an ARX-model (2.6.1), the prediction error can be written as:

$$\begin{aligned}\varepsilon(t) &= -A(q^{-1};\theta)y(t) + B(q^{-1};\theta)u(t) \\ &:= \varphi^T(t)\theta\end{aligned} \qquad (5.3.1)$$

which is of the equation error type.
The parameter θ may be computed directly from the normal equations (2.4.13). A direct Least Squares method for ARX-models, using the QR-decomposition instead of directly solving the normal equations, is implemented in the PRIMAL module DLS.

The Least Squares problem may also be solved in a recursive way, using the algorithm (5.2.17), and substituting $\psi(t) = \varphi(t)$. The recursive algorithm is utilised in Guidorzi's method, [Guidorzi, 1975] and is implemented in the PRIMAL module GUIDORZI, [Renes, 1984].

The estimation of the parameters in equation error models has the advantage of moderate computational cost and, more important, no numerical optimisation algorithms are needed, but instead the global minimum is found directly. However, in general the results will be biased, see (2.4.15).
Another cause of problems may be the emphasis put on fitting the high frequency part of the transfer function, which may result in a bad simulation model if only the initial Markov parameters are estimated correctly (see the previous section).
If, for instance, the process responses are slow compared to the sampling interval, an equation error method may find a good one step ahead prediction by extrapolating the values of past outputs. However, the output error will generally be large.

To prevent biasedness, the model may be extended with a noise filter $H(q^{-1};\theta)$. In this case the regression vector $\varphi(t)$ is no longer independent of θ and one has to resort to the general PEM.

FIR-model

For the FIR-model (2.6.2), the prediction error can be written as:

$$\varepsilon(t) = y(t) - B(q^{-1};\theta)u(t) \qquad (5.3.2)$$

In this case $\varepsilon(t)$ represents the output error. The parameters of $B(q^{-1};\theta)$ have a direct physical interpretation, being the finite impulse response parameters (i.e. Markov parameters) of the model.
As with the ARX-model, the Least Squares method is implemented in an direct way (module DLS) and in a recursive way (module MARKOV), see [Van der Linden & Renes, 1989e].

The estimation of the Markov parameters has several important properties.

- Provided the inputs are independent of the output noise, the parameters are asymptotically correctly estimated, irrespective of the noise properties.
- In the case of a persistently exciting input signal of sufficient order, the solution is unique.
- To get a good fit of the impulse responses of the process, the degree of $B(q^{-1};\theta)$ must generally be chosen high.
- The advantage of the high order is that the user is not confronted with difficult choices concerning order or structural indices. Furthermore, delays may be estimated within the model set as zero coefficients of $B(q^{-1};\theta)$.
  The method is therefore especially useful if little knowledge of the process behaviour is yet available.
- The large number of parameters makes the method unsuited for short datasets and for low-informative inputs.

If successful, the Markov parameter estimation produces a model showing good simulation behaviour. However, due to its high order the model is usually not immediately suited for its intended use.
Therefore, it should be reduced to a low order model by using realisation techniques, or, alternatively, it may be used as an initial estimate in the estimation of a lower order model, [Backx, 1987].
In PRIMAL, realisation methods based on the singular value decomposition of the Hankel matrix have been implemented, [Zeiger & McEwen, 1974; Damen & Hajdasinski, 1982; Kung, 1979; Gerlings, 1987].

The basic algorithm is given below:
Given a finite dimensional block Hankel matrix H, composed of Markov parameters. The singular value decomposition of this matrix is computed:

$$H = \begin{vmatrix} M_1 & M_2 & \cdot\cdot & M_k \\ M_2 & M_3 & \cdot\cdot & \\ \cdot\cdot & \cdot\cdot & \cdot\cdot & \\ M_k & \cdot\cdot & \cdot\cdot & M_{2k} \end{vmatrix} = U\,\Sigma_k V^T = (U\Sigma_k^{1/2})(\Sigma_k^{1/2}V^T) \qquad (5.3.3)$$

In order to approximate the high-order model with an n-dimensional state-space model, only the first n singular values are taken into account. Substituting $M_i = CA^{i-1}B$, the Hankel matrix can be written as the product of the observability and the controllability matrix.

The matrices of the state-space model can then be constructed with:

$$\begin{aligned} A &= (\Sigma_n)^{-1/2}\, U_n^T\, H_s\, V_n (\Sigma_n)^{-1/2} \\ B &= (\Sigma_n)^{1/2}\, V_n^T\, E_p \\ C &= E_q\, U_n (\Sigma_n)^{1/2} \\ D &= M_0 \end{aligned} \tag{5.3.4}$$

where $E_p^T = [\ I_p \mid 0\ ]$, $E_q = [\ I_q \mid 0\ ]$.

V_n, U_n are the first n columns of V, resp. U.

Σ_n is the upper left nxn matrix in Σ.

H_s is a shifted Hankel matrix, found by shifting all columns p places to the left. Its construction may be used to create several variants, see Damen & Hajdasinski [1982].
The realisation method leaves the user with the question which singular values are to be considered negligible. This problem may be tackled by inspecting the singular values and their ratio's, [Backx, 1987], and has been implemented in the PRIMAL module HANKDIM. It is experienced that in the case of "noisy" Markov parameters the decision may not be clear.
If the model is intended to show a good simulation performance, we may take a more suitable approach, by simulating with realisations of different dimensions, using the process data. The relative output error criterion, in combination with a penalty on the number of parameters (such as the AIC criterion), is used to select the appropriate dimension. The computational cost of this approach is moderate, since the singular value decomposition of the Hankel matrix just has to be computed once.

5.4 Instrumental variable methods

The Bootstrap IV, including a grid search for the delays, is implemented in PRIMAL as an iterative method in the module CIV, [Van der Linden & Renes, 1989e], using an instrument vector of delayed inputs and undisturbed outputs, computed by filtering the inputs by the process model obtained in the previous iteration step, cf. [Young, 1984].

The four-step (approximately optimal) IV will be briefly discussed below.

A four-step IV

Based on the model:

$$A(q^{-1};\theta)y(t) = B(q^{-1};\theta)u(t) + v(t) \tag{5.4.1}$$
$$D(q^{-1};\beta)v(t) = C(q^{-1};\beta)\epsilon(t) \tag{5.4.2}$$

taking the full polynomial form of (2.6.1), we obtain the corresponding regressions:

$$y(t) = \varphi^T(t)\theta + v(t) \tag{5.4.3}$$

$$\varepsilon(t) = \eta^T(t)\beta + v(t) \tag{5.4.4}$$

Consider the IV-method, with a matrix of instruments Z(t) and a stable data prefilter $F(q^{-1})$:

$$\hat{\theta} = [\sum_{t=1}^{N} Z(t)F(q^{-1})\varphi^T(\theta)]^{-1} [\sum_{t=1}^{N} Z(t)F(q^{-1})y(t)] \tag{5.4.5}$$

Under certain assumptions, including that the "true" system belongs to the model set, it can be proved that for any IV of this type, it holds that the asymptotic parameter covariance matrix:

$$P_{IV} \geq P_{IV}^{opt},$$

meaning that the difference matrix is non-negative definite, [Söderström & Stoica, 1983a, 1983b].

$$P_{IV}^{opt} = E\{[H^{-1}(q^{-1})\tilde{\varphi}^T(t)]^T \Lambda^{-1} [H^{-1}(q^{-1})\tilde{\varphi}^T(t)]\}^{-1} \tag{5.4.6}$$

Equality is achieved for the following choices of $F(q^{-1})$ and Z(t):

$$Z(t) = [\Lambda^{-1}H^{-1}(q^{-1})\tilde{\varphi}^T(t)]^T \tag{5.4.7}$$

$$F(q^{-1}) = H^{-1}(q^{-1})$$

where $\tilde{\varphi}(t)$ represents the noise-free counterpart of $\varphi(t)$, i.e. replacing the delayed outputs by the their corresponding undisturbed equivalents.

The optimal IV estimation cannot be applied directly, since knowledge of the $\tilde{\varphi}(t)$ and $H^{-1}(q^{-1})$ is required to filter the data and generate the instruments. To overcome this problem a multi-step algorithm has been proposed by Söderström & Stoica [1983a].

Step 1:
Apply an arbitrary IV-method to (5.4.3).
The resulting estimate will be denoted $\hat{\theta}_1$.

Step 2:
Apply a prediction error method to estimate β in
$y(t) = \varphi^T(t)\hat{\theta}_1 + H(q^{-1};\hat{\theta}_1,\beta)\varepsilon(t;\hat{\theta}_1,\beta)$.
Denote the result $\hat{\beta}_2$. In the MIMO-case estimate $\hat{\Lambda}_2$

Step 3:

Compute the optimal IV-estimate, as given by (5.4.7), using $\hat{\theta}_1$ to form $\tilde{\varphi}(t)$, and $\hat{\theta}_1$, $\hat{\beta}_2$ to form $H(q^{-1})$ and $\hat{\Lambda}_2$ to replace Λ. The resulting estimate will be denoted $\hat{\theta}_3$.

Step 4:

Repeat step 2 with $\hat{\theta}_3$.

This multi-step algorithm offers a certain amount of freedom in its construction. For the implementation in PRIMAL [Berben, 1987] the following choices were made. A full polynomial model was parametrised as, [Jakeman & Young, 1979]:

$$\theta = \text{col} [A_1^T, \ldots, A_{na}^T, B_1^T, \ldots, B_{nb}^T] \tag{5.4.8}$$

For the first step, a bootstrap IV technique which starts-up with a Least Squares method, has been implemented. For Step 2 a Pseudo Linear Regression (PLR) is used, [Stoica, 1984].

Remarks:

- The optimal accuracy of the IV-method is proven by Söderström & Stoica [1983a, b] under the restriction that the true system belongs to the model set. If this is not the case, no general statements can be made about the accuracy.

- In general it may be expected that an IV-method will perform less well than a corresponding PEM, [Söderström, 1989]. However, the IV-method is computationally more efficient, because the process and noise transfer function are estimated separately.

5.5 Effective new variants

As has been experienced, the identification methods discussed in the previous sections may perform badly if we are dealing with short datasets and a relatively high noise level, such that:

- The estimation of high-order models (such as the FIR model) will fail to deliver a satisfactory model.
- It is difficult to estimate possible delays.
- Equation error methods may not be appropriate.
- Estimation of low-order models may show bad convergence.

To find a suitable linear model, requiring a good simulation performance, we turn to output error methods. In practical implementations of output error methods, usually an equation error method is employed as a start-up method, optionally followed by an IV-method. However, the parameter vector at which the minimal equation error is found, may be far from the parameter vector minimising the output error criterion. As a result the output error method may have problems, showing bad convergence and/or convergence to a local minimum.

In these cases a more suitable start-up method, which also minimises the output error criterion, is more appropriate.

In this section we will discuss a start-up method based on the output error, which is robust, in the sense that it cannot diverge, and which is also well-suited to estimate the delays.
The principal idea is to combine a Monte-Carlo approach with Least Squares, optionally followed by a zero-order optimisation method.

A Monte-Carlo approach

Consider the MISO model:

$$y(t) = \sum_{i=1}^{p} \frac{B_i(q^{-1};\theta)}{F_i(q^{-1};\theta)} q^{-d_i} u_i(t) + v(t) \qquad (5.5.1)$$

where d_i denotes the delay.
We can separate this model into two components:

$$w_i(t) := F_i^{-1}(q^{-1};\theta)q^{-d_i} u_i(t) \qquad (5.5.2)$$

$$y(t) = \sum_{i=1}^{p} B_i(q^{-1};\theta)w_i(t) + v(t) \qquad (5.5.3)$$

Equation (5.5.3) is linear in the parameters and can be solved using a Least Squares method. We now introduce the following Monte-Carlo algorithm, cf. [Renes & Van der Linden, 1987]:

```
Step 1:
    Randomly select the required number of time constant values and
    delays from the space of feasible values.
Step 2:
    Compute the signal $w(t) = [w_1(t), w_2(t), \dots, w_p(t)]$,
    using (5.5.1).
Step 3:
    Minimise the prediction error criterion (5.2.1), using an
    ordinary Least Squares method.
Step 4:
    Compute the output error variance and accept the model if its
    value is smaller than the current value.
    Repeat from step 1, until a certain number of iterations is
    made.
```

This algorithm is strikingly simple, and it can be implemented very efficiently. Note that it has the desired property that it:
- cannot diverge,
- is capable of "escaping" from local optima,
- uses the same modelstructure as the output error PEM,
- and makes no assumptions concerning the noise properties.

The major drawback is, of course, that it may take a very large number of simulations to cover the parameter space sufficiently well. This is especially true if the number of parameters is large. Therefore the algorithm is suited only for estimating low-order models. It should be noted that under the conditions stated above, that is precisely what we want.

Simulation experiments show that in almost all cases, a few hundred iterations are sufficient to attain a solution near the optimum.
The Monte-Carlo approach performs well, owing to the Least Squares step:

- The LS method scales the impulse responses to each input to the appropriate ranges. If an input has a small effect on the output the LS step will reduce its influence, and consequently badly selected poles for this input have little effect on the performance.
- If the degree of the B-polynomial is sufficiently high, the LS step compensates badly selected poles by zeros.

The performance of the Monte-Carlo method can be further optimised by switching to a zero-order search method (e.g. the simplicial method, cf. [Box, Davies and Swan, 1969]). The method of Box finds the optimum of a function by moving and deforming a simplex in multi-dimensional parameter space, according to rules based on the function values in the vertices.
Zero-order methods have the advantage that mixed integer/real problems may be handled, so that the delays and parameters of the F-polynomials in (5.5.1) may be simultaneously optimised.

The procedure described above has been implemented in PRIMAL as a stand-alone method in module MCR, and as the start-up method of an output error PEM in module MCRPEM. See Chapter 6 for their performance in practice.

Remark:
Monte-Carlo and zero-order methods are not very popular in the field of identification. Typically, the literature on this subject dates back to the 1960's. However, as the results prove, the method produces results comparable to the other investigated identification methods in a computation time that is comparable to the high-order methods. Therefore we think that, taking into account the simplicity and robustness of the method, the method is especially useful in practical applications to get estimates for the delays and to give a first estimate of the achievable model performance.

CHAPTER 6 CASE STUDIES

6.1 Introduction

The Experimental Modelling strategy described in Chapter 3, and implemented in the PRIMAL package, has been applied to a variety of industrial processes.
In this chapter we discuss the application of PRIMAL in three cases:

case I	:	an industrial glass feeder process.
case II	:	a para-xylene crystallization process.
case III	:	a toluene-xylene distillation process.

For each case we will discuss the identification experiments, the results obtained in the identification and (in cases II and III) the performance of the control system.

Since (part of) the models were to serve feed-forward control, the output error was chosen to be the primary criterion of assessment in identification.
The results of the identification methods will be presented in tabular form. Each entry consists of:

a) The (PRIMAL) name of the method, see Chapter 5 and Appendix B. (real. : realisation with the HANKEL method, see Section 5.3).
b) The selected polynomials: A,B,C,D,F (see Section 2.6).
c) The type of method and model:

ee	:	equation error type PEM
oe	:	output error type PEM
ree	:	recursive equation error type PEM
roe	:	recursive output error type PEM
iv	:	instrumental variable method

d) The selected polynomial degrees: na, nb, nf
e) the relative output error (defined in Section 3.6) in the validation interval and cross validation interval.

6.2 Case I An industrial Glass Production Process

This section describes the application of PRIMAL to a Glass Production Process at Philips.

Process description

The process consists of a glass furnace followed by a feeder, cf. Figure 6.2.1. In a furnace, quartz sand is melted to produce liquid glass. A continuous stream of glass flows out of the furnace through a feeder. The main function of the feeder is to cool the glass down to a temperature suited for a shaping process. To satisfy the requirements for the shaping process the glass at the outlet of the feeder, called the spout, must be kept at a constant and homogeneous temperature profile. A detailed description of the process is given in [Van Vucht, 1987].

The control inputs of the feeder for this purpose are:
- The gas flow rate to the burners in the first section.
- The gas flow rate to the burners in the second section.
- The cooling air flow rate.

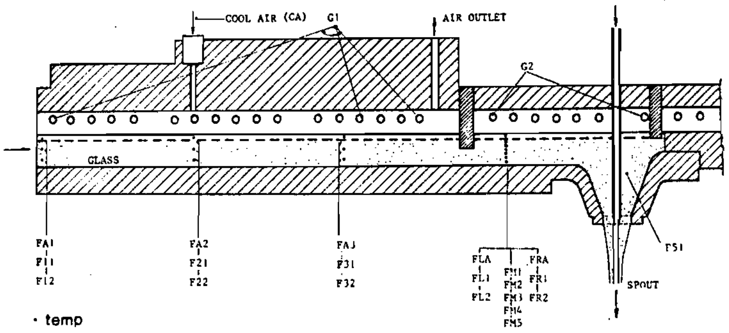


Figure 6.2.1:
Outline of the glass feeder. The glass enters the feeder at the left and leaves the feeder at the spout on the right.

The main process inputs and outputs are indicated by the following names:

G1:	measured gas input flow rate to section 1
G2:	measured gas input flow rate to section 2
CA:	measured cool air input flow rate to section 1
F51:	spout temperature (outlet temperature)

The purpose of experimental modelling is to model the dynamical behaviour of the temperature profile near the spout in response to the inputs listed above, in order to improve its control, see [Backx, 1987].

The most important dynamical relations are:

- The relation between the three control inputs and the glass temperature at the spout.
- The relation between the three control inputs and the homogeneity of the temperature distribution in the glass, as measured by thermocouples in a cross-section of the feeder (FL1, FL2, FM1, FM2, FM3, FM4, FM5, FR1, FR2).

The experimental modelling project was carried out in cooperation with and under supervision of the local systems and control group PICOS, at Philips. PRIMAL was used to supervise the design and to control the experiments, to monitor the experiment and to carry out the on-line analyses.

In the sequel, the results obtained with the various identification methods in PRIMAL are discussed. The analyses with PRIMAL were carried out in 1987, independently from and in parallel to the work of Backx [1987], who used an identification technique based on Finite Impulse Response (FIR) estimation and the subsequent estimation of a so-called MPSSM-model.

Instrumentation

The process serves as a pilot plant for testing new operating procedures and is therefore well-equipped with sensors. A transportable computer system taking care of analog data pretreatment, analog to digital conversion, and a front-end computer for real-time data acquisition, were coupled to the conventional instrumentation system of the process, cf. Figure 6.2.2.
A VAX/VMS computer was coupled to the front-end to perform the analyses of the measured data. In this second system the PRIMAL package was installed. A communication protocol was developed to exchange commands and experiment data between PRIMAL and the front-end in real-time.

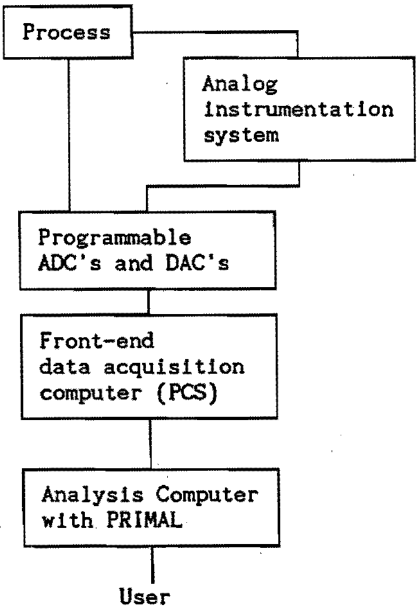

Process
Analog instrumentation system
Programmable ADC's and DAC's
Front-end data acquisition computer (PCS)
Analysis Computer with PRIMAL
User

Figure 6.2.2:
Schematic overview of the experiment set-up

Identification Experiments

On the basis of a priori knowledge and preliminary experiments, identification experiments were carried out involving small, independent PRBS, simultaneously imposed on the three inputs. About 45 output signals were measured with a sampling interval T_0 = 50 s. The PRBS had a clock period of 500 s.
12472 samples were collected during about 7 days of experiment time. In Figure 6.2.4, some of the collected signals are shown. Figure 6.2.3 lists a part of the experiment log book, kept automatically by PRIMAL.

```
20MAY87 11:25:03  SESSION BEGIN
20MAY87 11:26:25   #EDIT PRBNS130.EXP
20MAY87 11:31:03   #LOAD PICOS WITH PRBNS130
20MAY87 11:32:23   #PUT PRBNS(WIDTH=10,AMPL=40,SEED=4234)
                    ON PICOS.GAS_1_IN:10
20MAY87 11:32:24   #PUT PRBNS(WIDTH=10,AMPL=40,SEED=7645)
                    ON PICOS.GAS_2_IN:10
20MAY87 11:32:25   #PUT PRBNS(WIDTH=10,AMPL=40,SEED=26911)
                    ON PICOS.AIR_IN:10
20MAY87 11:32:48   #START PICOS
20MAY87 12:22:39  * Something goes wrong.
20MAY87 12:27:01  * There is a measurement problem with the gasflows.
20MAY87 12:28:31  * Also with cool_air sample range 10-30
20MAY87 13:57:11  * Remark of 12:27 irrelevant
20MAY87 13:57:11  * Cool_air corrected by Roel
21MAY87 11:12:06  * All signals inspected, cool_air shows some oscillation
                    near sample 1000. Also oscillation in FA6, F1_AIR and
                    F31.
21MAY87 11:19:27  * I'm going to look if the oscillations are visible after
                    filtering the data.
21MAY87 11:22:07   #APPLY FFT TO PICOS.DATA
21MAY87 11:23:09  * FFT was applied to raw data of F31, AIR_IN, COOL_AIR, F51
21MAY87 11:23:09  >>Application FFT;001 stopped
21MAY87 11:54:27   #APPLY PREFILTER TO PICOS.DATA
21MAY87 11:56:43  >>Application PREFILTE;001 stopped
21MAY87 12:01:59   #APPLY FFT TO PREFILTER
21MAY87 12:02:17  >>Application FFT;002 stopped
21MAY87 12:12:00  * F2_GAS and F2_AIR get stuck at the low side
21MAY87 12:48:35  * On sample 1817 the fysical offset of F2_AIR, F2_GAS
                    raised from 20% to 70%.
21MAY87 14:36:48  * On sample 1936 F2_AIR, F2_GAS reset by PUNIC-card
                    adjustment.
21MAY87 14:40:03   #APPLY CORRELATOR TO PREFILTER
21MAY87 14:40:30  >>Application Correlat;001 stopped
21MAY87 14:40:30  * First attempt to see correlation with the spout F51
...
21MAY87 15:24:40   #APPLY PREFILTER TO PICOS.DATA
21MAY87 15:40:23   #APPLY CORRELATOR TO PREFILTER
21MAY87 15:43:22   #APPLY MARKOV TO PREFILTER
21MAY87 15:44:54   #STOP PREFILTER
...
21MAY87 15:54:27  * Not much to be seen yet
22MAY87 09:09:56  * Operator remarks: 7.30u (sample 3100) low spout
                    temperature caused high spread in tube diameter.
22MAY87 09:25:28   #APPLY PREFILTER TO PICOS.DATA
22MAY87 09:57:06   #APPLY CORRELATOR TO PREFILTER
22MAY87 10:11:58  * First correlation results,
                    F1_GAS*FA3 acceptable, COOL_AIR*FA3 absent??, ...
...
```

Figure 6.2.3:
Part of the original log book kept by PRIMAL.
(The remarks have been translated into English).

The on-line signal analyses were mainly used to monitor the experiment and check the existence of correlation between the inputs and the important output signals.

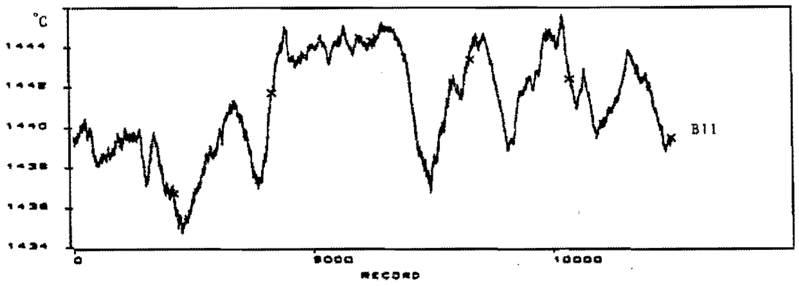


Figure 6.2.4(a):
Glass temperature in the furnace preceding the feeder.

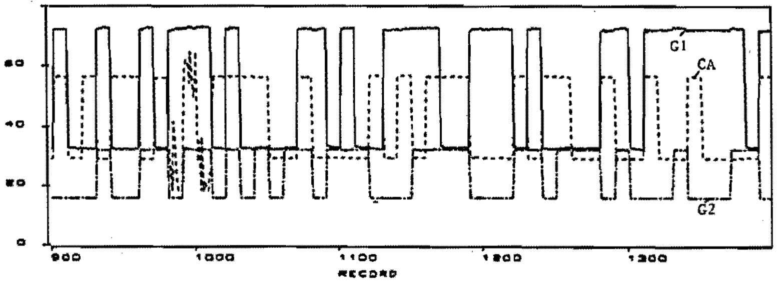


Figure 6.2.4(b):
Part of the PRBS on G1, CA and G2.

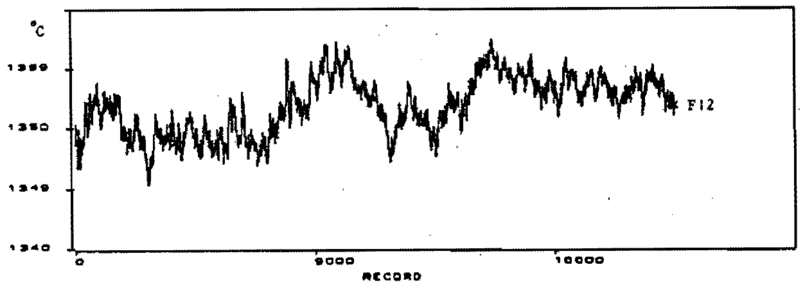


Figure 6.2.4(c):
Glass temperature F12 in the beginning of the first section.

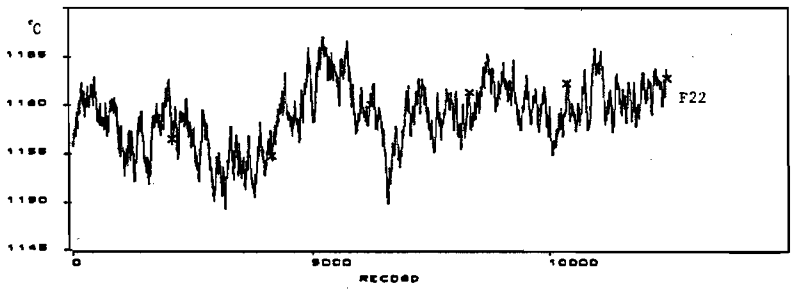


Figure 6.2.4(d):
Glass temperature F22 in the middle of the first section.

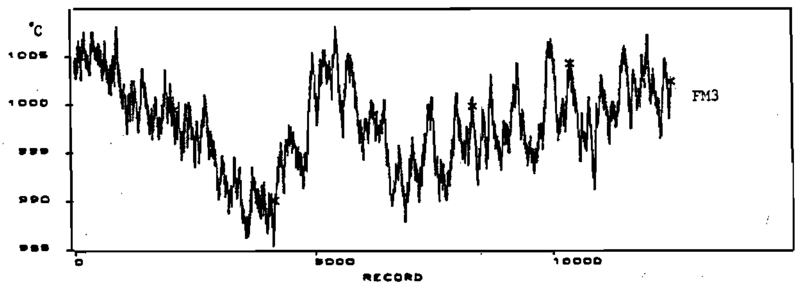


Figure 6.2.4(e):
Glass temperature FM3 in second section.

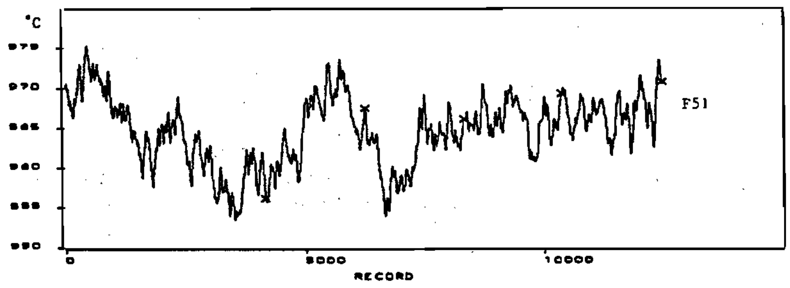


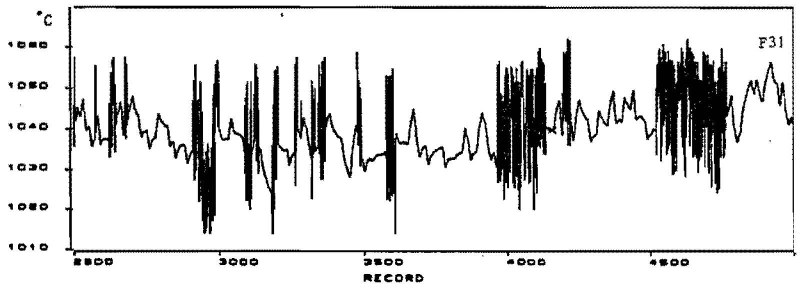
Figure 6.2.4(f):
Spout temperature F51.

Data conditioning

The data conditioning consisted of two steps. In the first step the data were corrected for measurement errors, such as outliers.
Several output signals were disturbed severely by outliers. Figure 6.2.5 illustrates the effect of automatic outlier correction using the method described in Chapter 3.
The inputs to the model were measured signals (C1, CA, C2), which were corrected for the sensing delays.
After the outlier and delay correction, the power spectra of the inputs and outputs were computed to determine the bandwidths of the measured signals.
The results are shown in Figure 6.2.6 for three representative signals. As is clear from this picture, the sampling rate is high enough for all dynamical interactions, and the clock frequency of the PRBS is sufficiently high for all glass temperatures.

The results of the first data conditioning step indicate that the data may be reduced by a factor of 10 (in the time). To achieve this, the data is filtered with a sharp (high-order) low-pass FIR-filter to eliminate all frequencies higher than $1/(5*T_0)$ Hz. After application of this digital anti-aliasing filter, the excess data is reduced by selecting 1 sample out of every 10 samples.

Before removing trends in the data that were not caused by the PRBS, but by slow changes in the furnace and the environmental temperature, the cross correlation functions of the control inputs and the temperatures were computed.
These functions were used to get a first impression of the delays and time constants involved, and further to check the causality of the input-output relations.
The results of the correlation analyses for some signals, are shown in Figure 6.2.7:
- (a) gives an impression of the impulse responses, since the inputs are nearly white. As expected, the effect of C2 is much faster than C1 and CA.
- (b) the correlation functions of the inputs and the output temperatures in the cross-section of the second section are all similar.
- (c),(d) the correlation functions of C1, CA and the temperatures at the end of the first section are nearly identical, with opposite sign, which implies that their effect is probably due to the same physical principle. This result indicates that it may be difficult to control the vertical temperature gradient in the glass.



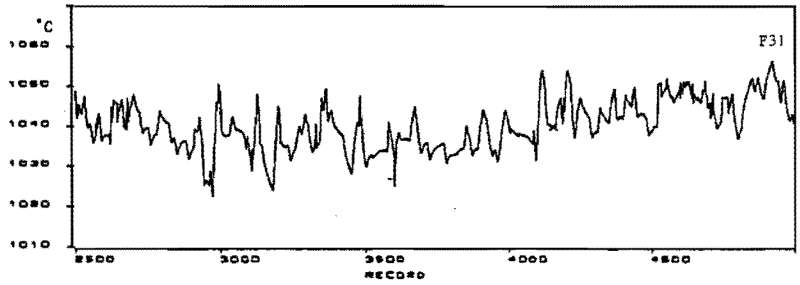
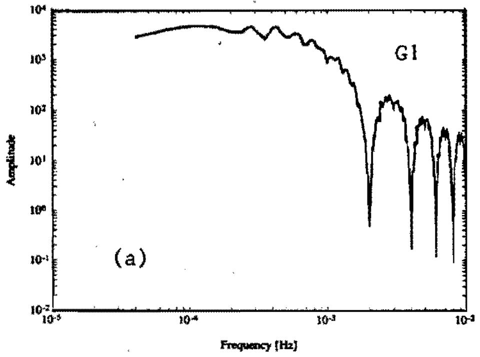
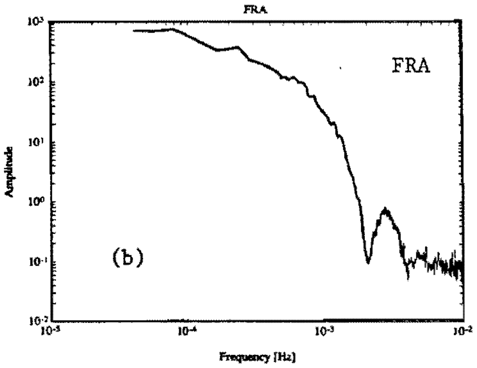


Figure 6.2.5:
(a) A part of signal F31 corrupted by outliers.
(b) F31 after automatic outlier correction with the median filter, see Section 3.3.4.





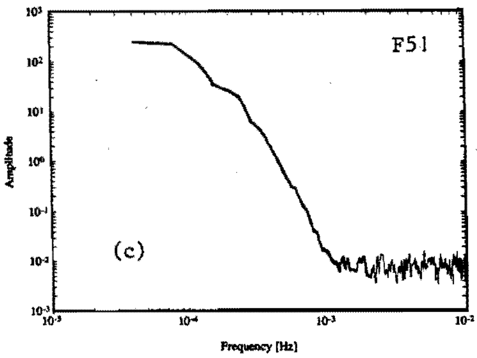


Figure 6.2.6: Spectra of (a) G1, (b) FRA and (c) F51.

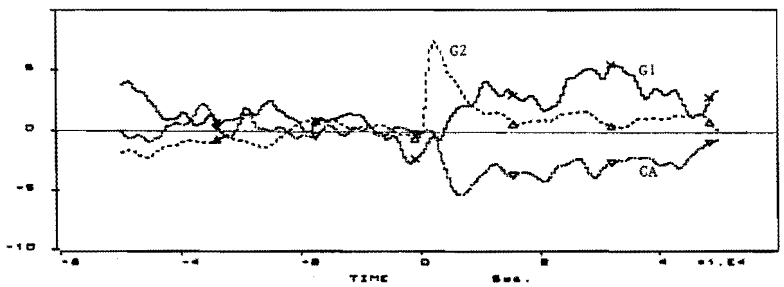


Figure 6.2.7(a): Cross-correlations between the inputs and F51.

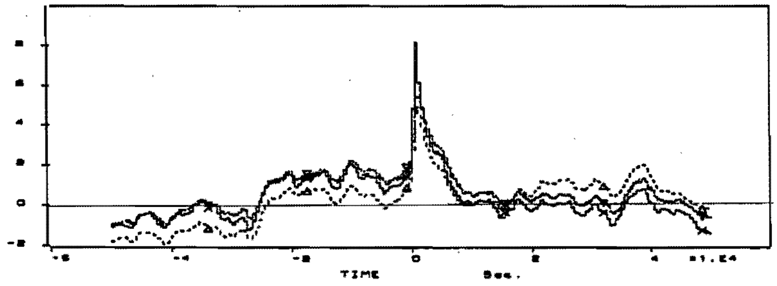


Figure 6.2.7(b): Cross-correlations between G2 and temperatures in a cross-section of section 2.

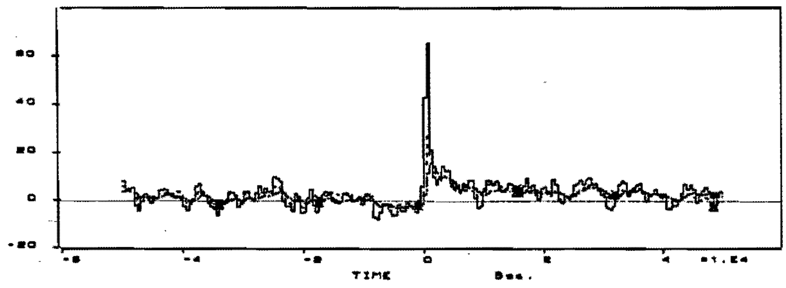


Figure 6.2.7(c): Cross-correlations between G1 and FA3, F31 and F32.

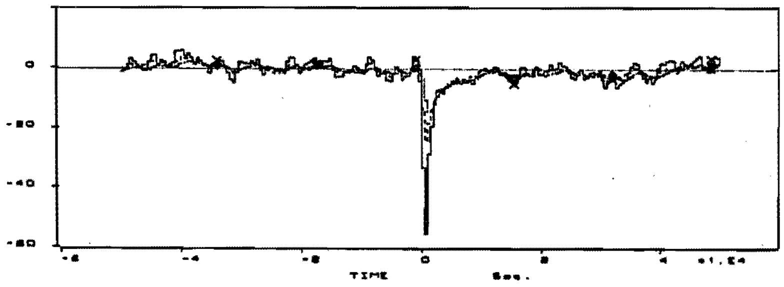


Figure 6.2.7(d): Cross-correlations between CA and FA3, F31 and F32.

Identification results

After conditioning the data, various identification methods in PRIMAL have been applied, all using samples 1..750 of the (reduced) dataset. To evaluate the performance of the methods, the output error was computed, under equal conditions, over the identification interval (1..750) and over a cross-validation interval (750..1247). The results are shown in Table 6.2.8.

	method		type	nf na nb	relative output error (mre) validation	cross-validation
1	RPEM	BF	roe	5 5	3.1 %	4.8 %
2	CUIDORZI	BA	ree	8 8	2.7 %	3.7 %
3	real. n=3				3.2 %	3.2 %
4	MARKOV	B	roe	0 50	2.3 %	5.4 %
5	real. n=3				2.5 %	4.1 %
6	DLS	B	oe	0 50	2.2 %	5.8 %
7	real. n=3				2.5 %	4.2 %
8	DLS	BA	ee	8 8	2.7 %	3.7 %
9	CIV	BA	iv	6 6	2.7 %	3.2 %
10	MCR	BF	oe	3 3	2.1 %	3.8 %
11	IVM	BACD	iv	5 4	1.9 %	3.4 %
12	MCRPEM	BF	oe	3 2	1.6 %	3.6 %
13	MCRPEM	BF	oe	4 4	1.4 %	4.1 %

Table 6.2.8:
Representative results of different identification methods in modelling F51 on a validation and cross-validation interval.

Table 6.2.8 reveals that all identification methods achieve a small output error, which indicates that the process can be described accurately by a linear model and that the signal-to-noise ratio is excellent.
The lowest output errors on the identification interval are obtained by the output error method MCRPEM (Method 12, 13) and the four-step IV-method IVM (Method 11), using the delays estimated by MCRPEM. The impulse responses estimated by these methods are presented in Figure 6.2.9. As is the case here with all methods, the differences between the impulse responses are small. The output error methods arrive at a lower output error by modelling a longer tail of the impulse responses of CI and CA. This low-frequency component of the impulse responses proves to be artificial, causing a slightly higher output error in the cross-validation.

Remarks about the methods:

- Impulse response estimation works well and may be used to provide first estimates of the delays and the model orders.
  In the realisation step a third-order model proves to be sufficiently accurate.
- The Monte-Carlo method (MCR) provides good estimates of the delays and obtains a good model, usually within 100 trials. Its performance and efficiency is comparable to the other methods.
- Because the errors are small, the best models found with the equation error techniques show practically the same impulse responses as the models found by the IV- and output error techniques.

Figure 6.2.10 presents the fit of Model 11, see Table 6.2.8, on the validation and the cross-validation interval. The output error has mainly low-frequency components and is uncorrelated with the inputs.

In the identification of the transfer functions to the temperatures in a cross section of the glass bed (see Figure 6.2.1), similar impulse responses and accuracies were obtained. The results will therefore not be presented here. Backx reports similar results, obtained with the MPSSM-approach, see [Backx 1987, 1989].

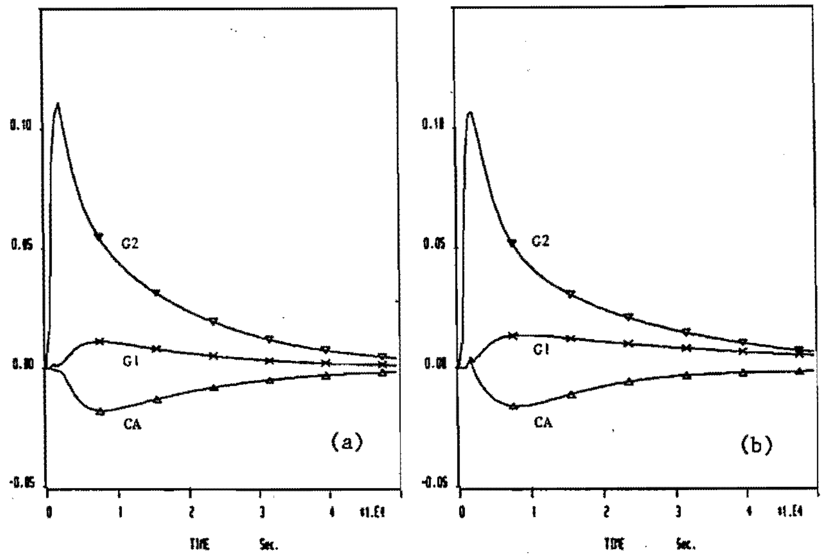


Figure 6.2.9:
(a) Impulse responses of Model 11 (estimated by IVM)
(b) Impulse responses of Model 12 (estimated by MCRPEM)

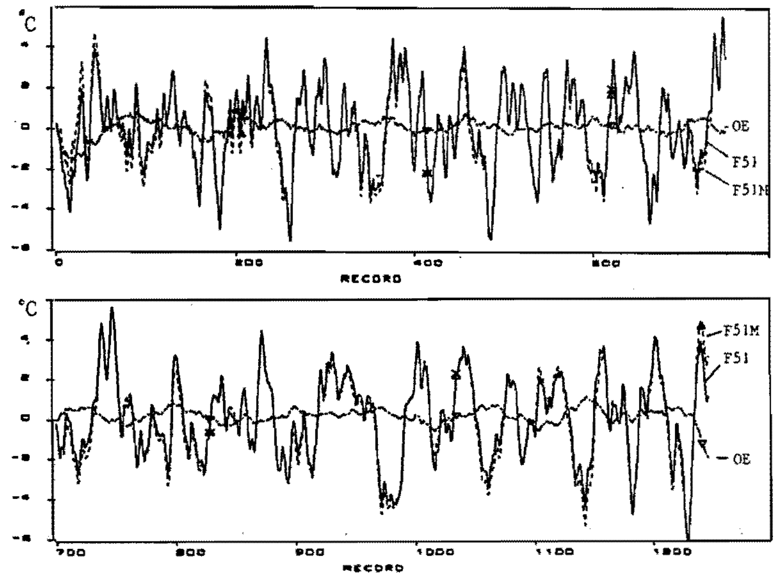


Figure 6.2.10:
Fit of Model 11 on the (a) cross-validation interval, (b) validation interval. F51M: model output, OE: output error

Conclusions

The dynamical behaviour of the output temperature F51 of the feeder process, which - on the basis of the physical principles - should be regarded as a highly non-linear, distributed parameter system, could be approximated well with a third-order linear black box model. In the identification dataset about 96 % of the energy in the output signal is explained by the model.
Owing to the excellent signal-to-noise ratio, and the conditioning of the data, all methods attain a similar output error level.
Comparison of the impulse responses shows that the small differences between the results of the methods are mainly found in the low-frequency part (tail) of the impulse responses.

Looking back upon the modelling exercise as a whole, we may further conclude, that:

- The impulse responses provide information about the delays in the responses of the spout temperature to the inputs.
- More surprisingly, the time constant values are very large compared to residence time of the glass, leading to the conclusion that the storage of energy in the wall of the feeder was a much more important factor than was expected.
- Remarkably and unexpectedly, the cool air flow rate and the energy input by the gas burners in the first section show comparable dynamics, indicating that the cool air input decreases the radiation temperature of the wall of the feeder.

Here we see but a few examples of how black box methods may provide valuable physical ("white box") insight.

6.3 Case II A Para-Xylene Crystallization Plant

This section describes the application of PRIMAL to an industrial crystallization plant.
The para-xylene production plant of EXXON Chemical Holland in Rotterdam isolates para-xylene from a mixture of para-, ortho- and meta-xylene, toluene and benzene. The process makes use of the high melting point of pure para-xylene (13 °C) with respect to the melting points of the other components (-95 to -25 °C).

Process description

The process consists of several stages of drums and centrifuges, see Figure 6.3.1. In each stage a slurry flow enters a set of centrifuges, where it is separated in a cake flow (consisting mainly of para-xylene crystals) and a liquid flow, called the filtrate.

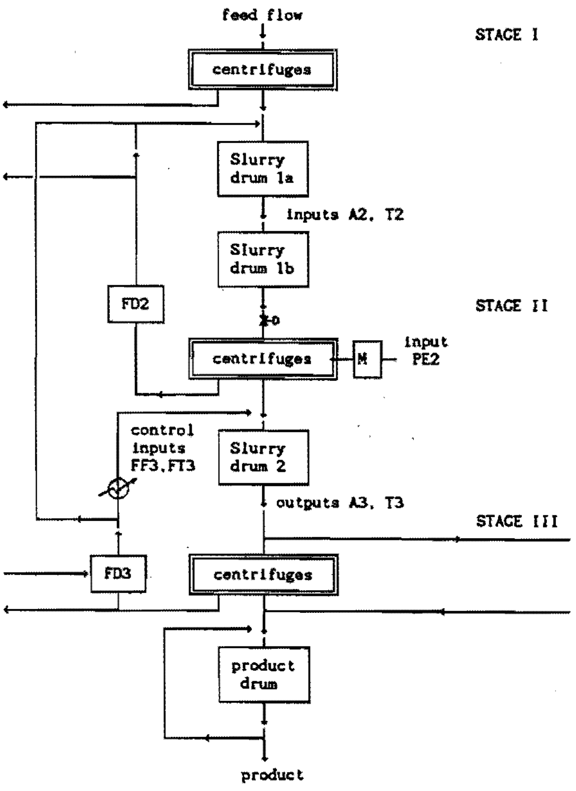


Figure 6.3.1:
Schematic overview of part of the Para-Xylene Crystallization Plant. FD = filtrate drum, SD = slurry drum.

The investigation of the process dynamics has concentrated on the second stage. A better control strategy had to be developed for the purity and solid content of the flow leaving the second-stage slurry drum (SD2). The temperature of the flow is taken as a measure of its purity and the goal is to keep this temperature (T3) near its set point, while maximising the solid content (A3) of the flow.

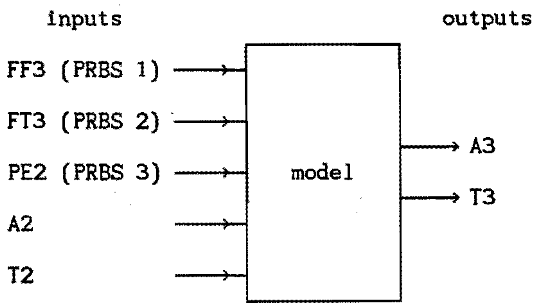
The solid content and temperature of drum SD2 are controlled by a filtrate flow of liquid para-xylene, that is recycled from the third stage centrifuges and that is heated in a heat exchanger. The control inputs are:
- the filtrate recycle-flow rate (FF3).
- the temperature of the filtrate flow (FT3).

A serious complication was that the flow rate, temperature and solid content of the cake flow entering the second stage slurry drum could not be measured. To cover this problem, the power required by the centrifuges (PE2) was taken as an indirect measure of the flow. The temperature (T2) and solid content (A2) of the slurry flow just after the first-stage slurry drum were taken as measures of the temperature and solid content.

On the basis of an investigation of the available process knowledge and a physical model based on first principles, cf. [Hogendoorn, 1988], experiments were planned to find the dynamical responses of A3 and T3 to:
- the 2 control inputs FF3 and FT3.
- the 3 inputs A2, PE2, and T2.

Identification Experiments

During the experiments the process had to stay within the specified range, so only small test signals were allowed. On the basis of a priori knowledge and preliminary experiments, PRBS were superimposed upon the set points of the slave controllers of FF3 and FT3. To get information about the dynamical influence of variations in the cake flow, a PRBS was also superimposed on a valve controlling the slurry flow to the second-stage centrifuges. The effect of the flow variations is measured by PE2.



The purpose of the identification experiment was to generate data for modelling the output A3. The test signals were designed to cause approximately equivalent variations in this output. To get accurate information about the dynamics, minimal PRBS clock periods of 5 minutes and 10 minutes were taken. 14 signals were measured with a sampling interval of 60 seconds.

During the experiment, signal analyses and identification methods were used to get preliminary models and to check the information contents of the data. It was decided to reduce the amplitude of the test signal in FT3 and to eliminate the test signals after 48 hours. Subsequently, 17 hours of normal plant operation were measured. Figure 6.3.2 gives an overview of the experiment data. Figure 6.3.3 presents their spectra. In order not to reveal proprietary information no physical ranges of the inputs and outputs are presented.

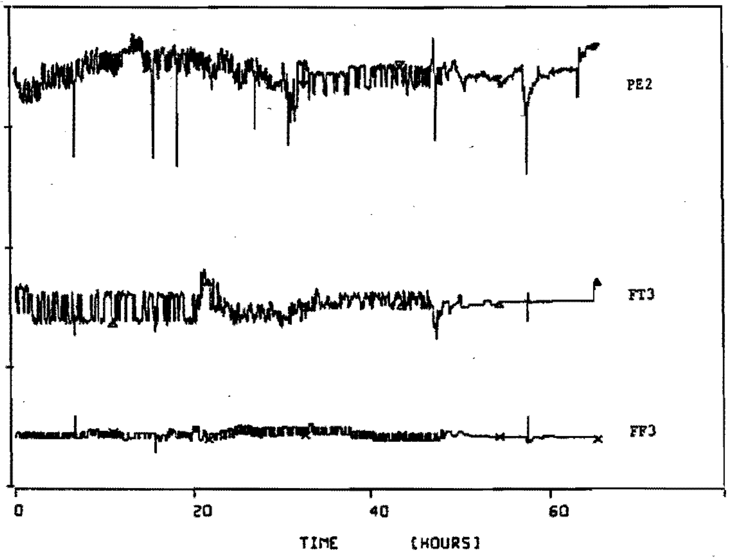


Figure 6.3.2 (a):
The inputs with the test signals: FF3, FT3 and PE2.

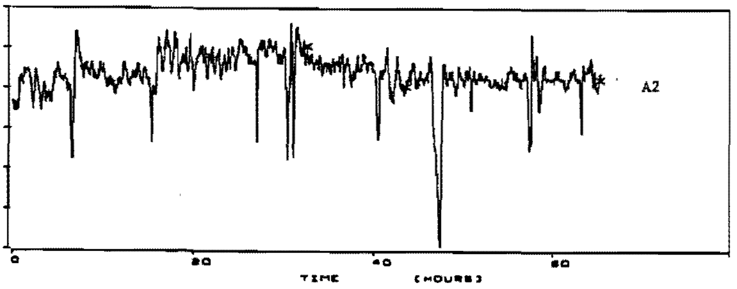


Figure 6.3.2(b): The solid contents A2 in the first stage.

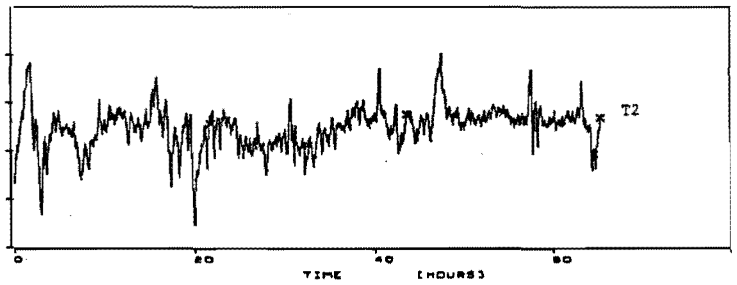


Figure 6.3.2(c): The temperature T2 in the first stage.

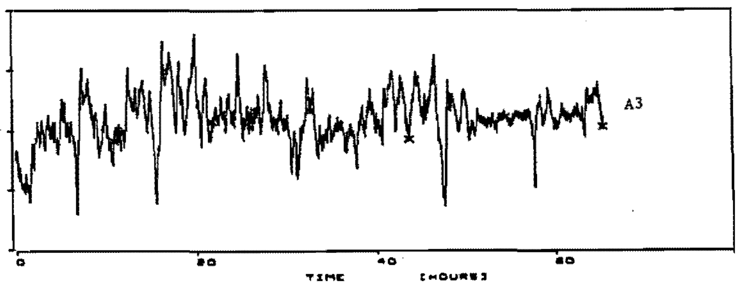


Figure 6.3.2(d): The solid contents A3 after slurry drum 2.

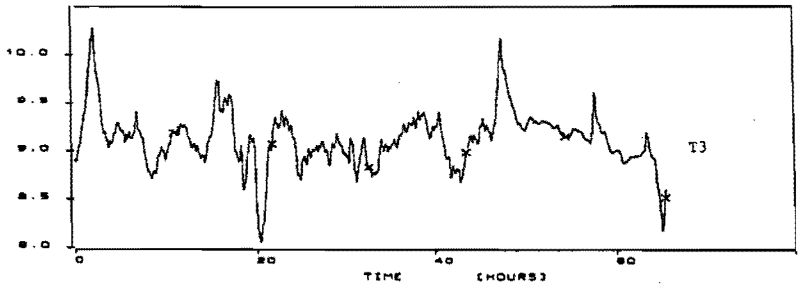


Figure 6.3.2(e): The temperature T3 after slurry drum 2.

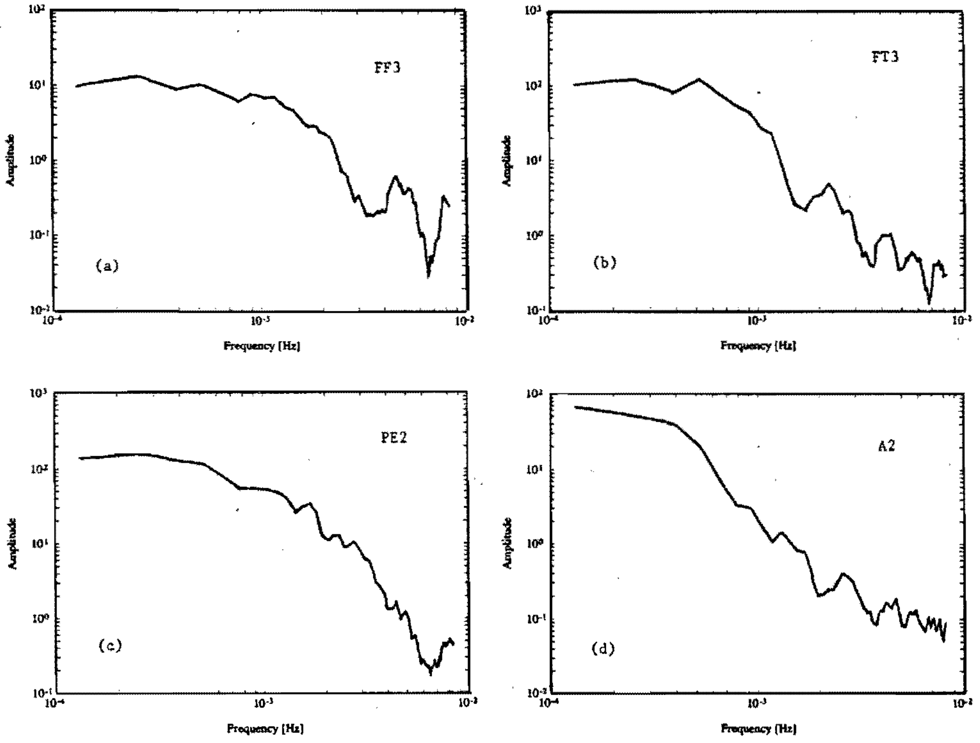


Figure 6.3.3 (a)-(d):
Spectra of (a) FF3, (b) FT3, (c) PE2, (d) A2.

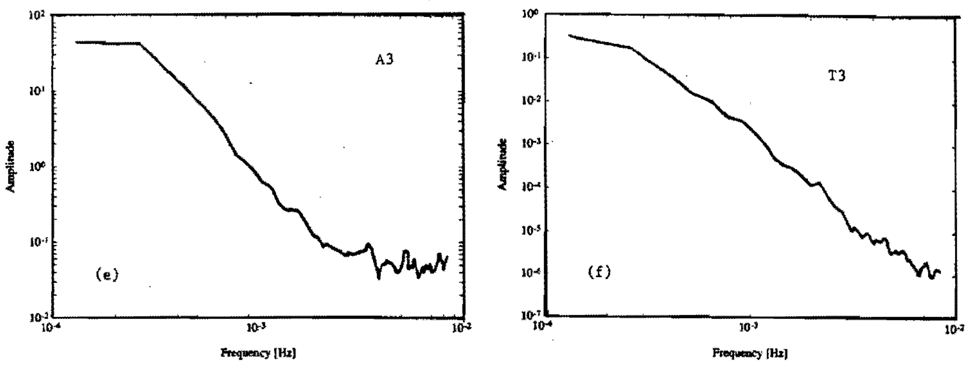


Figure 6.3.3 (e-f):
Spectra of (e) A3 , (f) T3.

Data conditioning

The data were corrected for measurement errors, measurement delays and trends. Analyses of the resulting data and of a dataset that was reduced to a sampling interval of 5 minutes, showed comparable results.

Identification results

A variety of identification methods in PRIMAL have been applied to the data. All methods have used 35 hours (samples 1..450 of the reduced dataset) for identification. To evaluate the performance of the methods, the output error was computed, under equal conditions, over the identification interval and a cross-validation interval of 28 hours (samples 450..786), of which 17 hours without test signals.

Results for the solid content A3

Initially, the identification was carried out for the three inputs with the PRBS test signals. This resulted in an output error of about 45 % . Correlating the output error with the other inputs showed that the solid content A2 had a significant influence on the behaviour of A3, reducing the output error to about 15 % , which immediately indicated the importance of taking the disturbances in the first stage into account.

Representative results for various identification methods in modelling A3, using FF3, FT3, PE2 and A2 as inputs, are shown in Table 6.3.4.

	method		type	nf na nb	relative output error (mre) validation	cross validation
1	RPEM	BF	roe	2 2	14 %	16 %
2	CUIDORZI	BA	ree	5 5	13 %	18 %
3	MARKOV	B	roe	0 15	12 %	19 %
4	real. n=2				16 %	17 %
5	MCR	BF	oe	1 1	15 %	18 %
6	DLS	BA	ee	4 4	14 %	17 %
7	CIV	BA	iv	5 5	14 %	14 %
8	MCRPEM	BF	oe	1 1	14 %	17 %
9	IVM	BACD	iv	4 4	13 %	14 %
10	MCRPEM	BF	oe	2 1	13 %	16 %
11	DLS	B	oe	0 15	12 %	18 %
12	real. n=2				15 %	17 %
13	MCR	BF	oe	2 2	11 %	21 %
14	MCRPEM	BF	oe	2 2	10 %	19 %
15	DLS	B	oe	0 20	10 %	19 %

Table 6.3.4:
Representative results of various identification methods in modelling the solid content A3, using inputs FF3, FT3, A2 and PE2.

Table 6.3.4 indicates that various identification methods show a comparable performance. As expected, the output error methods, estimating a high order impulse reponse model (DLS) or a low order model (MCR, MCRPEM), realise the lowest output errors. However, in the cross validation the best results were obtained by the models estimated with the IV-methods.
Figure 6.3.5 shows the simulated output and observed output of the model found by the four-step IV-method (IVM) on the identification and the cross validation interval.

Closer inspection of the models reveals that all methods find approximately the same impulse responses, see Figure 6.3.6.
Given sufficient freedom, the output error methods find a better fit to the data by adding low-frequency components to the impulse responses, as can be observed in the tail of the responses (see Figure 6.3.6 (c)-(d)). However, in the cross validation these low-frequency components prove to be artificial.

Observations:

a) Inspection of the results of the equation error methods show that the equation error is nearly white. A close correspondence of the estimates of equation error methods and output error methods may thus be expected.

b) The impulse responses show the expected behaviour, an increase of the 'hot' filtrate flow FF3 or an increase of its temperature FT3 results in a decrease of the solid content. A larger cake flow (PE2, A2) results in an increased solid content. The time constant values lie in the range of 10-20 minutes, corresponding to the mean residence time of the liquid in the drum.

c) Correlating the output error with the inputs reveals that no significant correlation remains. Thus, first or second order models are sufficient to describe the process' dynamics.

d) In the identification interval, the relative contribution of the inputs to the power of the output signal was FF3: 30%, FT3: 23%, A2: 29% and PE2: 18% .
During the period without test signals the model "explains" 85 % of the output signal and shows a relative contribution of the inputs to the output of 19 %, 4 %, 39 %, 38 % respectively.
These results show the importance of taking A2 and PE2 into account when developing a new control strategy.
Figure 6.3.7 shows the contribution of each input to the model output. Obviously, A2 causes the large drops in A3.

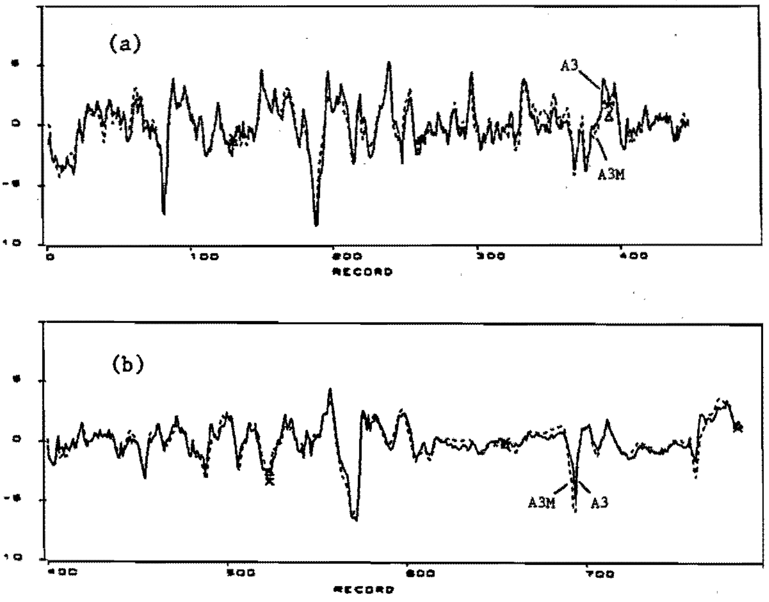


Figure 6.3.5:
Observed output A3 and simulated output A3M of Model 9: (a) validation interval, (b) cross-validation interval.

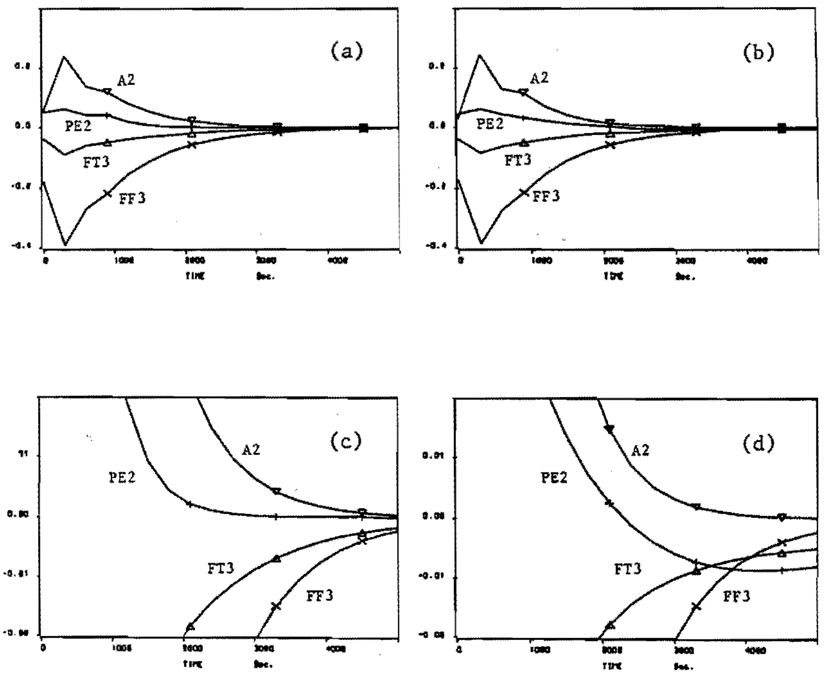


Figure 6.3.6:
(a) Impulse responses of Model 9 (estimated by IVM).
(b) Impulse responses of Model 10 (estimated by MCRPEM).
(c) Tail of the impulse responses of Model 9.
(d) Tail of the impulse responses of Model 10.

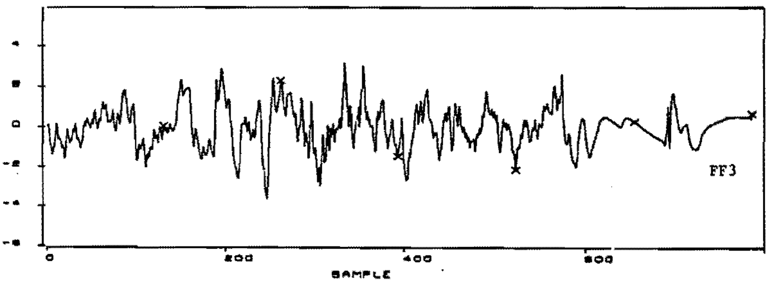


Figure 6.3.7(a): Contribution of FF3 to A3 (total dataset).

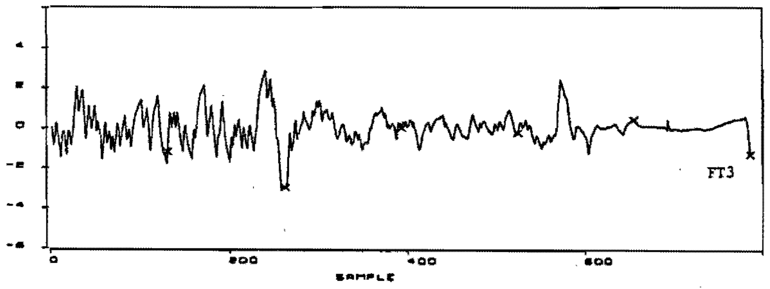


Figure 6.3.7(b): Contribution of FT3 to A3 (on the total data).

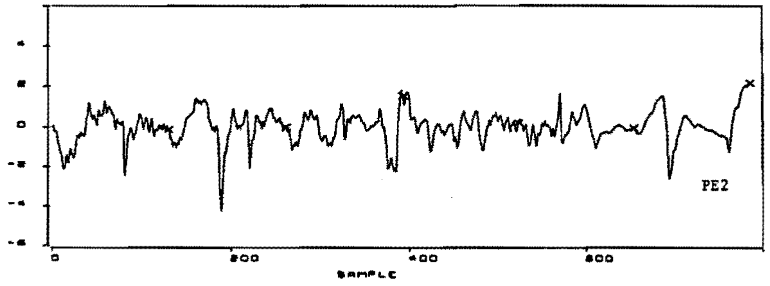


Figure 6.3.7(c): Contribution of PE2 to A3 (on the total data).

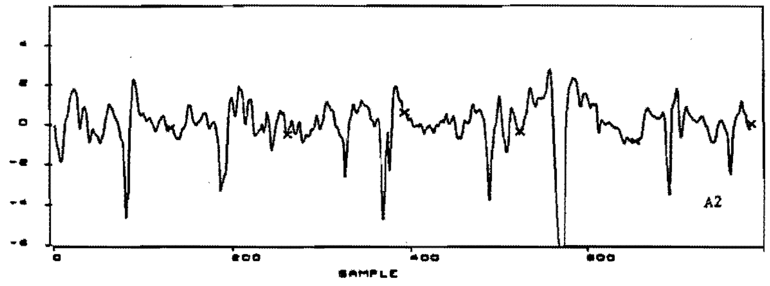


Figure 6.3.7(d): Contribution of A2 to A3 (on the total data).

Results for the temperature T3

Although the identification experiment was intended to model the solid content (A3), the intermediate analyses of the data indicated that a satisfactory model could also be found for the temperature T3.
The results showed a negligible influence of A2, but a large contribution of T2 to T3. Therefore, models were estimated using FF3, FT3, PE2 and T2 as inputs.
Representative results for T3 are presented in Table 6.3.8.

				nf		relative output error (*mre*)	
	method		type	na	nb	validation	cross-validation
1	GUIDORZI	BA	ree	6	6	24 %	32 %
2	RPEM	BF	roe	2	2	21 %	26 %
3	MARKOV	B	roe	0	20	20 %	33 %
4	real. n=3					33 %	33 %
5	IVM	BACD	iv	3	3	27 %	33 %
6	DLS	BA	ee	6	6	25 %	33 %
7	CIV	BA	iv	4	4	24 %	29 %
8	MCRPEM	BF	oe	2	2	22 %	29 %
9	MCRPEM	BF	oe	2	3	21 %	26 %
10	MCRPEM	BF	oe	2	4	19 %	31 %
11	DLS	B	oe	0	20	17 %	32 %
12	MCR	BF	oe	3	3	16 %	38 %

Table 6.3.8:
Representative results of various identification methods in modelling T3, using inputs FF3, FT3, PE2 and T2.

The output error methods obtain the best results in the validation and cross validation interval.
Comparing the impulse responses reveals that the response to T2 is nearly identical for all methods. The responses to FT3 and PE2 differ slightly and the response to FF3 varies considerably.
This behaviour can be explained by looking at the relative power contribution of the inputs to the output: FF3: 3 %, FT3: 23%, PE2: 18%, T2: 56 % . Due to its small contribution, the response to FF3 is not estimated reliably. See Figure 6.3.9, for the impulse responses of Model 9, estimated by MCRPEM.
The output error methods estimate a longer tail in the impulse responses of PE2 and FT3, compared to the IV- and equation error methods. This results in an improved fit of the low frequency disturbances in both the validation and the cross validation interval.
The simulated output of model 9 in Table 6.3.8 is presented in Figure 6.3.10. For this model the contribution of each input to the model output is presented in Figure 6.3.11.

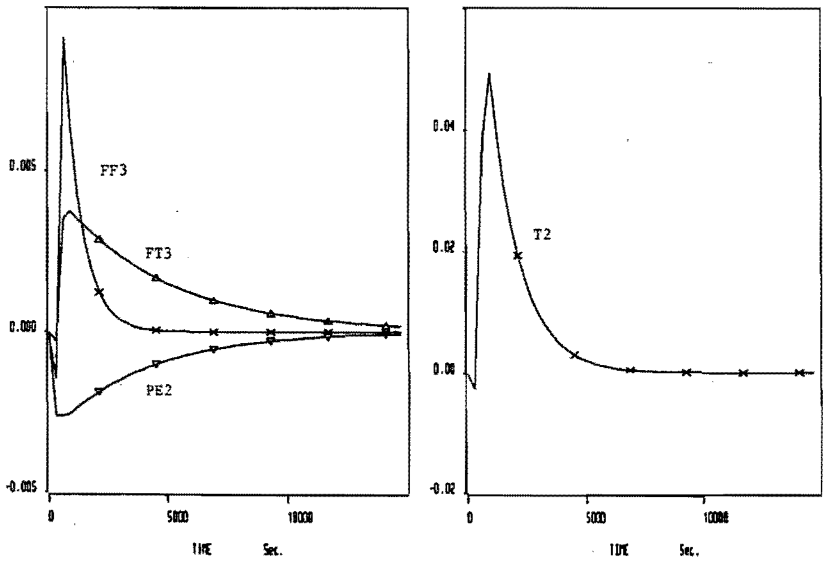


Figure 6.3.9: Impulse responses of Model 9.

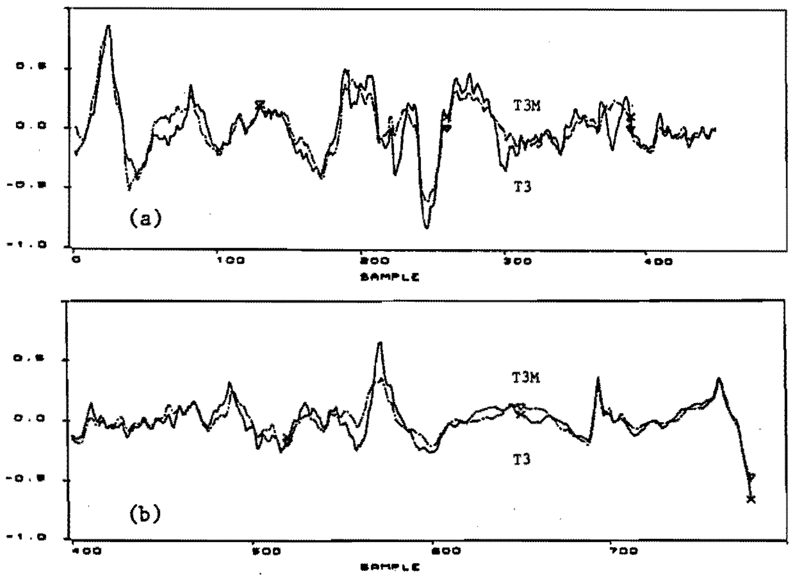


Figure 6.3.10: Observed output T3 and simulated output T3M, by model 9 in the (a) validation interval, (b) cross validation interval.

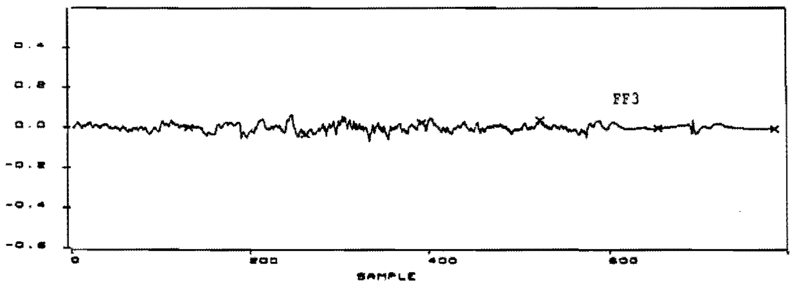


Figure 6.3.11 (a): Contribution of FF3 to T3 (total data set).

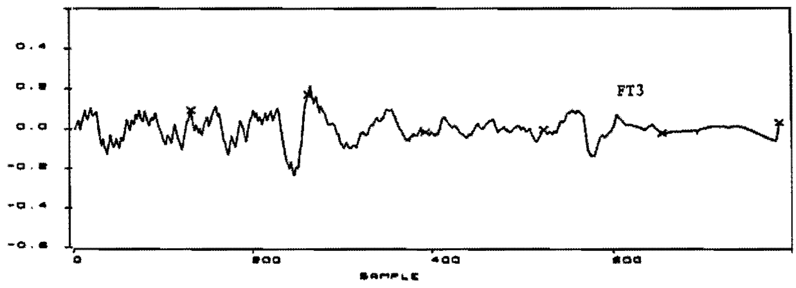


Figure 6.3.11 (b): Contribution of FT3 to T3 (total dataset)

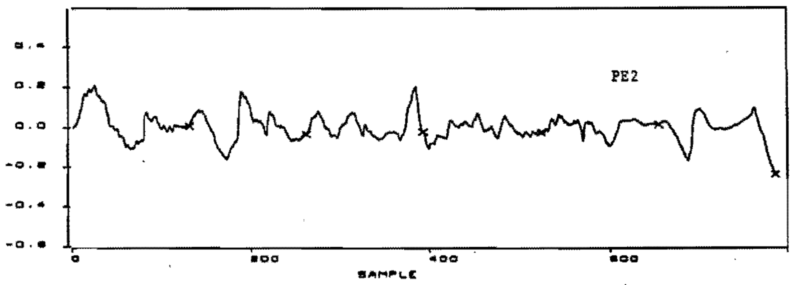


Figure 6.3.11 (c): Contribution of PE2 to T3 (total dataset).

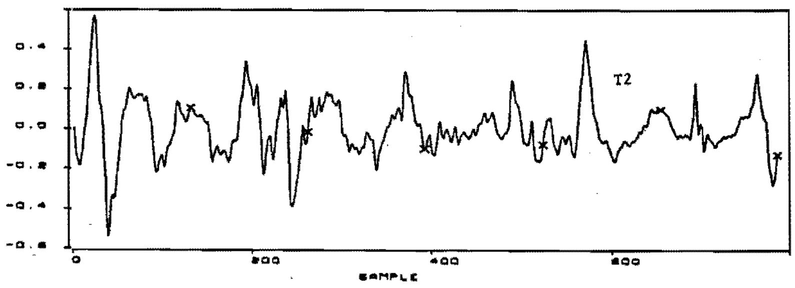


Figure 6.3.11 (d): Contribution of T2 to T3 (total dataset).

Observations:

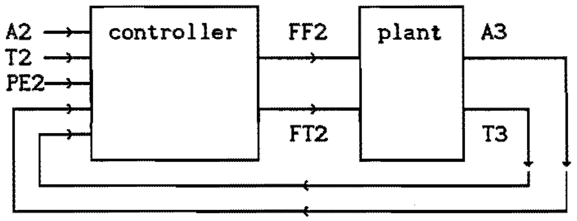
a) The responses show that an increase of the temperature of the filtrate flow (FT2) or the cake flow temperature (T2) leads to a delayed increase of T3. An increase of the (cold) cake flow rate (PE2) has the expected negative influence on T3. Increasing the hot filtrate flow rate (FF2) leads to an increase of T3, although this effect is small compared to the effect of the other inputs. The time constants lie in the range of 20 minutes to 1 hour, which was larger than expected.

b) The impulse response estimation results in "noisy" Markov parameters due to the short identification interval compared to the length of the response.

c) The four-step IV-method has problems with instability of the estimate in the third step in several cases.

Control system design

The parameters of the estimated first-order models were used in the implementation of a "dynamically reconciled" control system, cf. [Bartman, 1981], where disturbances in T2, A2 and PE2 are compensated for by feed-forward. The remaining disturbances are taken care of by the feed-back loop.



The performance of the controller was monitored during eight days and compared to the performance during conventional operation, see Figure 6.3.12.
The new control system reduced the standard deviation of A3 and T3 by 40 % and 30%, respectively.

Conclusions

The identification results show that the process dynamics of the second stage of the para-xylene production process can be modelled sufficiently well, using only first-order and second-order models of the SISO subsystems.

The analyses have shown that the indirect measurements of the slurry flow (PE2) and the temperature and solid content (T2, A2) in the first stage show a strong correlation with the behaviour of the output temperature and solid content (T3, A3) in the second stage. Moreover, the large disturbances in A2 and T2, (caused by "wash" operations to prevent plugging in the first stage), were a major cause of control problems in the second and third stage.
By estimating the dynamical influence of disturbances in A2, T2 and PE2, these measurements could be used in feed-forward control. The controller developed for the second stage takes A2, T2 and PE2 into account, which results in a significantly improved control performance.

The results clearly indicated that an even better performance might be achieved by reducing the number of wash operations, for instance by changes in the plant for reducing plugging, e.g. by heating the walls of the pipes and drums.

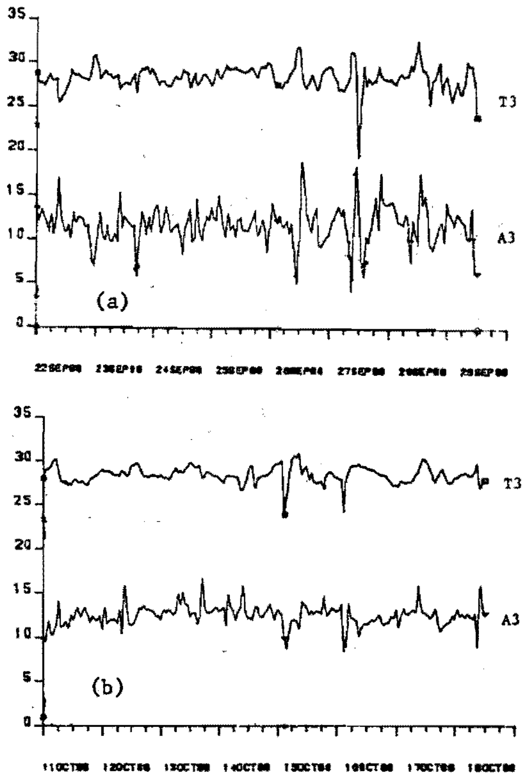


Figure 6.3.12:
Behaviour of T3 and A3 during two eight day periods:
(a) with conventional control.
(b) with the new control system.

6.4 Case III A Toluene-Xylene Distillation Plant

This section describes the application of PRIMAL to a distillation unit for separating toluene from xylenes at EXXON Chemical Holland in Rotterdam. The unit is part of a fractionation section comprising three distillation towers. The bottom product of two parallel benzene distillation towers, consisting of a mixture of benzene, toluene and xylenes, constitutes the feed flow to the Toluene Tower: its top product (toluene) is delivered to tankage and its bottom product (xylene) is further treated in the Xylene Plant.

Process description

A schematic description of the toluene-xylene tower is presented in Figure 6.4.1.

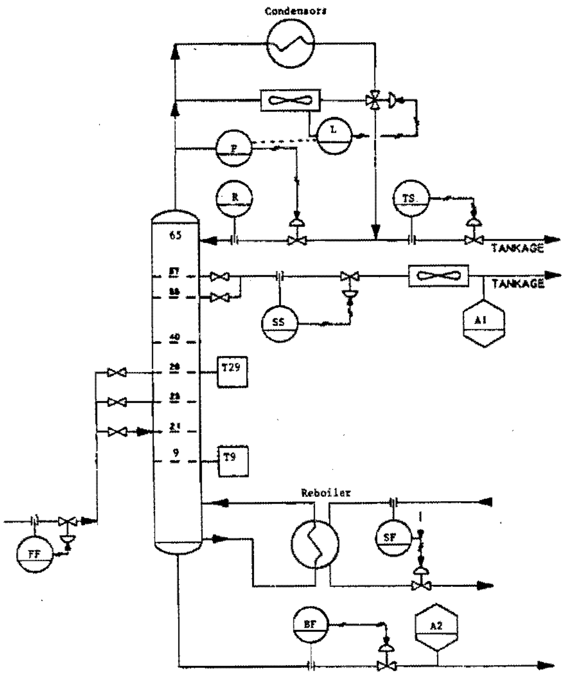


Figure 6.4.1:
Schematic overview of (part of) the Toluene-Xylene Distillation Plant.

The tower has 65 trays. The feed enters at tray 21. The toluene product leaves the tower as a "side-stream" at tray 55 or 57. A small flow consisting of benzene and toluene is extracted at the top and is recycled (through tankage) to the beginning of the fractionation section.
The heat input to the tower is delivered by a steam-heated flooded reboiler. The top vapour flow is condensed and cooled by an air-cooled condensor in parallel to a flooded condensor, which serves heat-integration with other distillation units.

The main goal is to utilise the maximum capacity of the tower, while keeping the xylenes content of the side-stream product on specification (maximally e.g. 800 ppm xylenes). The quality requirement for the bottom product is less critical: the flow may contain maximally e.g. 1.5 % toluene.

The compositions of the side-stream and bottom products are measured by analysers (A1 and A2), delivering new values every 15 minutes.
If the tower operates at a fairly constant pressure, the faster responding temperatures in the lower part of the rectifying section (T29) and in the stripping section, near the bottom (T9), may be taken as indicative measures of the product compositions.

As control inputs we consider:

- the reboiler steam flow rate (SF).
- the side-stream flow rate (SS).
- the reflux flow rate (R).

Disturbance sources are:

- the weather, which influences the rate of heat transfer in the condensors.
- variations in the feed flow rate (FF) and feed composition (AF).

The maximum capacity at cool weather is determined by the steam flow rate.

The experiments were planned to find the dynamical responses of T29 and T9 to variations in:

- the feed flow rate: FF.
- the control inputs: SF, SS and R.

During the experiments the tower was under top pressure control by the reflux flow rate, see Figure 6.4.1. To get an impression of the dynamical influence of the reflux, small zero-average reflux variations were induced by variations in the top-stream flow rate: TS.

During the investigation of the process, early estimates of the process dynamics were made, making use of a simulation program of the plant, historical data and practical experience, cf. [Tolboom, 1989]. Only a few initial experiments could be carried out before planning the identification experiments. During all experiments, the top product composition had to stay within its specified range, so only small test signals were allowed.

Identification experiments

In the main identification experiment, PRBS were imposed simultaneously on the feed flow rate (FF), the side-stream flow rate (SS), the steam flow rate (SF) and the top-stream flow rate (TS), cf. Figure 6.4.2.

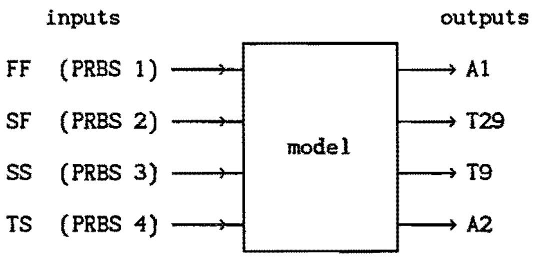


Figure 6.4.2:
Inputs and outputs of the model.

Since control of the top product composition is considered to be the most important, the identification experiment concentrated on modelling T29.
On the basis of the initial experiments and a priori knowledge, a minimal PRBS clock period of 10 minutes was selected. The input/output data was collected with a sampling interval of 30 seconds.

The experiment was carried out under what turned out to be rather unfavourable conditions, because the feed composition changed considerably, i.e. from 65 % to 87 % toluene, see Figure 6.4.4(f). To accommodate this change, the set points of the feed and the side-stream flow controllers had to be adjusted by the operators, see Figure 6.4.4(a). The top product has been off-spec twice during the experiment, see Figure 6.4.4(b).

Inspection of the experiment data revealed that the variations in the steam flow rate dominated in the responses of T29. Therefore, the amplitudes and periods of the PRBS in FF and SS were increased somewhat, whereas the amplitude of the PRBS in SF was reduced by a factor of three for the last 7 hours of the experiment.
For technical reasons the analyses had to be carried out off-line.

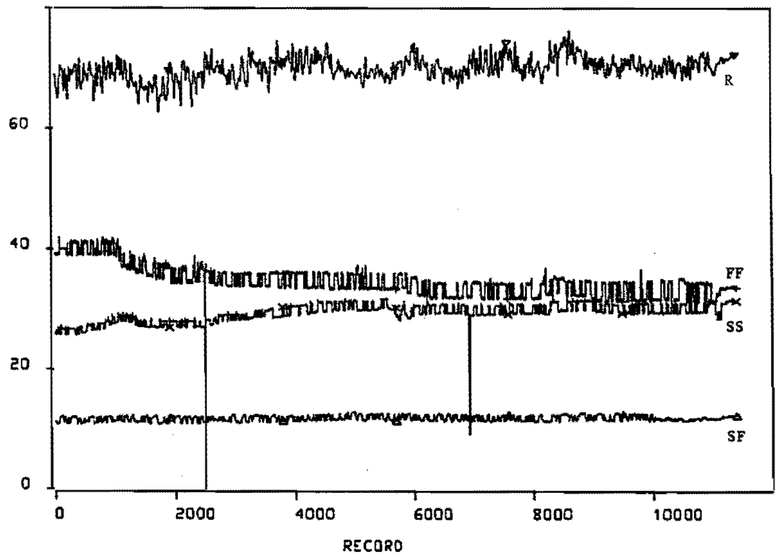


Figure 6.4.4 (a):
Reflux (R) [m^3/h], feed flow rate (FF) [m^3/h], side-stream flow rate (SS) [m^3/h], and steam flow rate (SF) [ton/h] as a function of the sample number (sampling interval: 30 s).

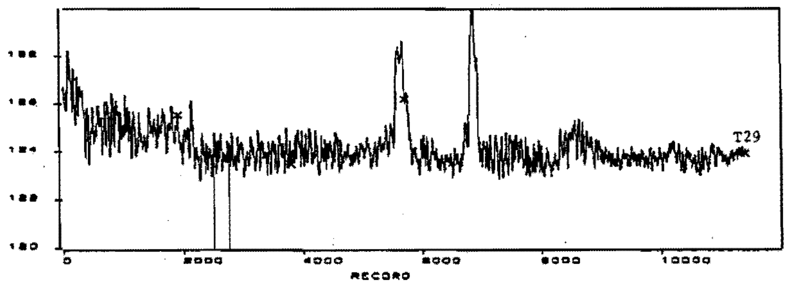


Figure 6.4.4 (b):
Temperature [°C] at tray 29 (T29).

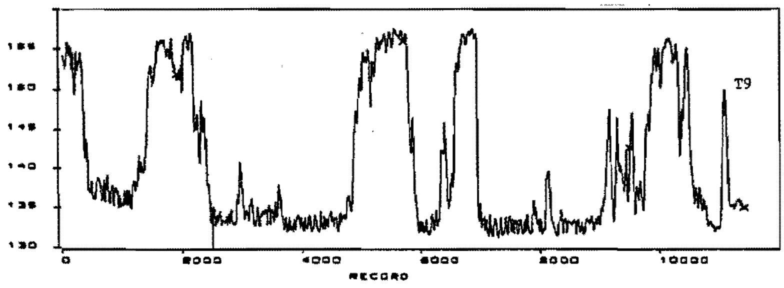


Figure 6.4.4 (c):
Temperature [°C] at tray 9 (T9).

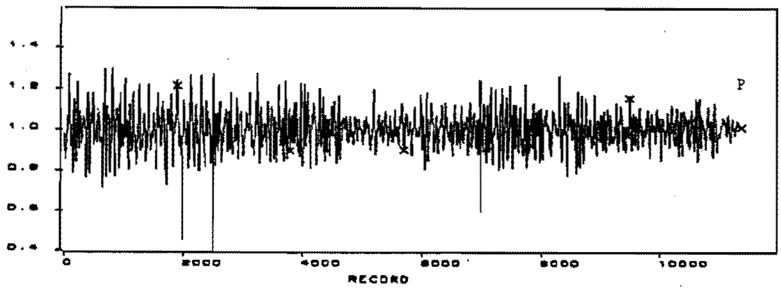


Figure 6.4.4 (d):
Pressure [bar] in the top of the tower (P).

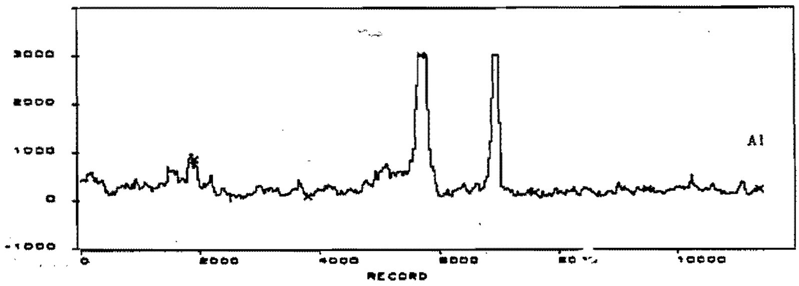


Figure 6.4.4 (e):
Xylenes-content [ppm] of the side-stream product (A1).

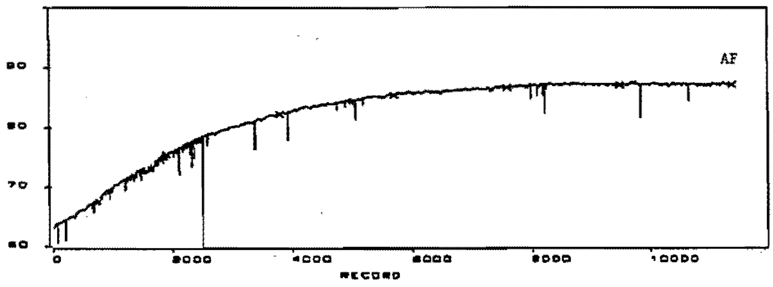


Figure 6.4.4 (f):
Toluene content [%] of the feed (AF).

Data conditioning

The data were corrected for outliers, measurement delays and trends. After applying a digital anti-aliasing filter, the data were reduced to sampling intervals of 2.5 minutes and 10 minutes.

Identification results

All identification methods have used 17 hours of the dataset, (samples 7000..9000). In this interval the feed composition was fairly constant. To evaluate the performance of the methods, the output error was computed over the identification interval as well as over a cross-validation interval of 21 hours, (samples 2500..5000). By selecting these intervals, the off-spec peaks in the side-stream composition were not taken into consideration.

Results for T29

The initial analyses showed that the top stream PRBS did not contribute significantly to the variations in T29, and therefore the model interrelating these two variables could not be estimated with sufficient accuracy. However, its dynamical influence could be determined from a separate reflux test, involving a test signal in the top stream only.
Subsequently, the remaining three inputs were used in the identification. Representative results of the various identification methods are listed in Table 6.4.3.

				na		relative output error (mre)	
	method			nf	nb	validation	cross validation
1	CUIDORZI	BA	ree	4	4	16 %	25 %
2	RPEM	BF	roe	2	2	16 %	24 %
3	MARKOV	B	roe	0	15	14 %	26 %
4	real. n=3					17 %	25 %
5	DLS	BA	ee	4	4	18 %	24 %
6	DLS	BA	ee	6	6	17 %	25 %
7	CIV	BA	iv	4	4	17 %	24 %
8	IVM	BACD	iv	4	4	16 %	25 %
9	MCR	BF	oe	2	2	16 %	23 %
10	MCRPEM	BF	oe	1	3	14 %	22 %
11	DLS	B	oe	0	15	14 %	24 %
12	MCRPEM	BF	oe	2	3	13 %	24 %

Table 6.4.3:
Representative results of various identification methods in modelling the temperature T29, using inputs FF, SF and SS.

As can be seen in Table 6.4.3, the differences in performance between the methods are small, both in the identification and in the cross validation interval.
In the validation and cross validation, the estimated models show a similar output error behaviour, the output error methods finding the best fit at the low frequencies, resulting in a smaller error variance.

The impulse responses of Model 10 to SF and FF are presented in Figure 6.4.5 (a). Due to the small contribution of the PRBS in SS to the variations in T29, the response of T29 to SS could not be estimated reliably.
Figure 6.4.7 presents the observed output (T29) and the predicted output on the validation interval and a data interval of 33 hours (samples 1000..5000), including the cross validation interval. As expected, a bad fit is found for the initial phase (samples 1000..2200), due to the shifting operating conditions.

The estimated model of the process dynamics incorporates the pressure control. As follows from Figure 6.4.4 (d) and the estimated impulse response of P to the steam flow SF, see Figure 6.4.5 (b), the pressure is not at all kept constant by the pressure controller. Therefore, the column temperatures not only respond to composition changes, but also to the changes in the pressure. The fast positive initial response of T29 to SF is caused by this effect.

Figure 6.4.6(a) shows the step responses of T29 to the inputs.
After the initial pressure effects, the temperatures show the expected slow response due to the composition changes. Remarkably, T29 did not respond to changes in SS in this experiment, while the side-stream composition, measured by the analyser A1, clearly did.
We did not yet have the opportunity to investigate this effect, which may be due to the position of tray 29 with respect to the composition profile in the tower.

Results for A1

For modelling the slow composition changes of the side-stream product, models have been estimated for the analyser output (A1), using SF, SS and FF as inputs. The step responses of A1 to the inputs, see Figure 6.4.6 (b) clearly demonstrate the expected very slow responses of the side-stream composition to step changes in the inputs, cf. [Rademaker, Rijnsdorp en Maarleveld, 1975]. As expected, a larger feed or steam flow rate increases the purity of the side-stream, while a larger side-stream flow rate decreases it.

Results for T9

The identification experiment focused on modelling T29, causing relatively large variations in T9, see Figure 6.4.4(c). During two periods, of 19 hours (samples 2460..4700) and 17 hours (samples 7000..9000), the temperature was in its normal range. Therefore,

these intervals have been used to estimate models of the bottom temperature, although influences of the preceding upsets might still be present.
Figure 6.4.8(b) shows that the faster variations in T9 can be largely explained by SF, and the slower variations by FF.

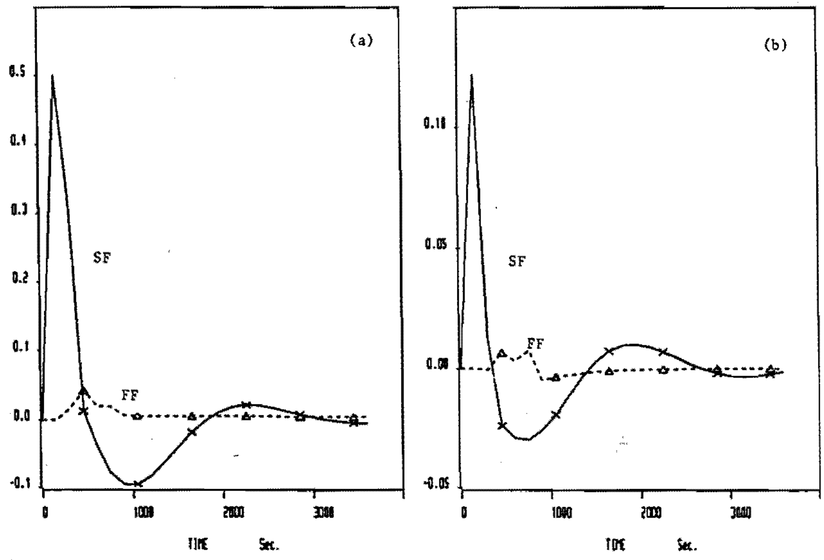


Figure 6.4.5:
(a) Impulse responses of T29 to FF [°C.h/m^3] and SF [°C.h/ton].
(b) Impulse responses of P to FF [bar.h/m^3] and SF [bar.h/ton].

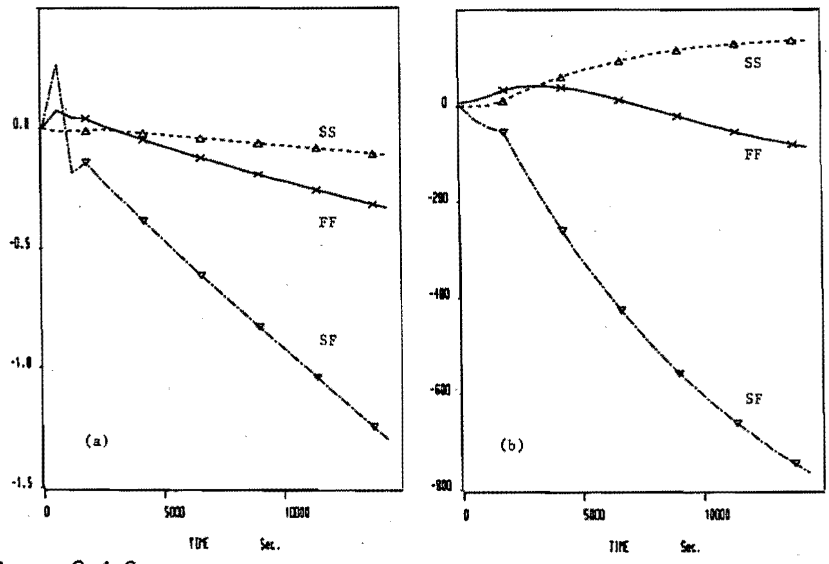


Figure 6.4.6:
(a) Step responses of T29 to FF, SS [°C.h/m^3] and SF [°C.h/ton].
(b) Step responses of A1 to FF, SS [ppm.h/m^3] and SF [ppm.h/ton].

Control

The estimated models were used to adjust the parameters of the feed-forward and feed-back in the plant's existing control scheme, which required first-order-plus-delay models of the dynamics.
The control scheme consists of:

1) feed-forward compensation for changes in feed flow rate and feed composition (ratio control),
2) feed-back of the temperatures T29 and T9, corrected for pressure changes, and:
3) feed-back of the top and bottom composition measured by the analysers,

all being combined to act upon the setpoints of the slave flow controllers of SF and SS.
The performance of the improved control scheme was monitored during 6 weeks, in which the energy demand decreased by 3 %, enabling a higher throughput during cool weather, when the heat input is the limiting factor.
Unfortunately, a better evaluation of the control performance could not be carried out, due to problems with a leaking valve in the steam circuit, which hampered the control by the heat input.

Obviously, the existing circumstances have not yet allowed a better exploitation of the potential possibilities for improving the control performance. For instance, a better performance may be achieved by implementing a better model of the dynamical influence of the side-stream, by improving pressure control and the correction for pressure changes in T29 and T9, and by adapting the control system, so as to make more effective use of the models than is possible with first-order-plus-delay transfer functions in the feed-forward, feed-back, and decoupling paths.

Conclusions

In this case, the experiment data were analysed off-line using PRIMAL. Satisfactory models have been found for the dynamical response of A1 to FF, SS and SF and for T29 to FF and SF. Pressure control turned out to be far less satisfactory than was thought before and during the experiments, and, unexpectedly, the influence of SS on T29 was much smaller than anticipated.
Looking back upon the modelling exercise, more complete information about the process behaviour would have been obtained if we had been able to carry out the analyses in real-time. This would have enabled us to detect the unexpected effects in an early stage and to modify the experiment accordingly.

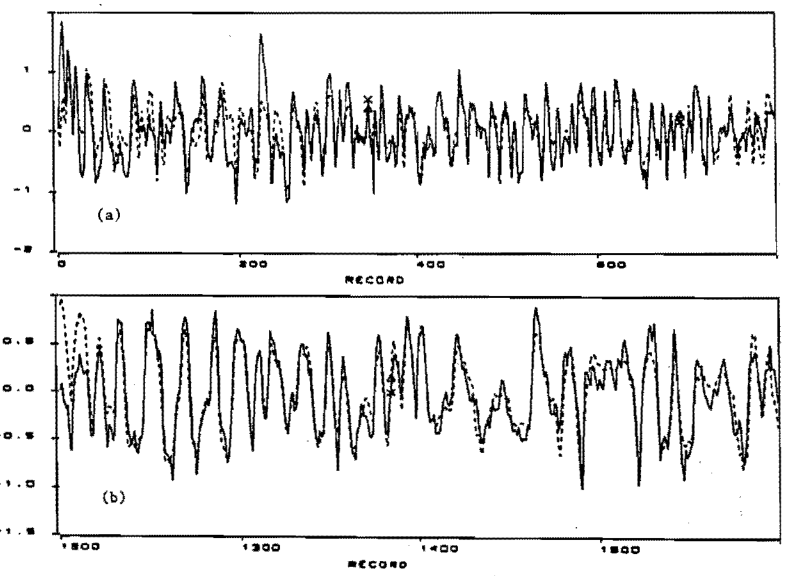


Figure 6.4.7:
Performance of Model 11 in validation and cross validation.
T29 [°C] "——" and model output T29M [°C] "— — —" on:
(a) an interval (samples 1000..5000) including the cross validation interval.
(b) the validation interval.

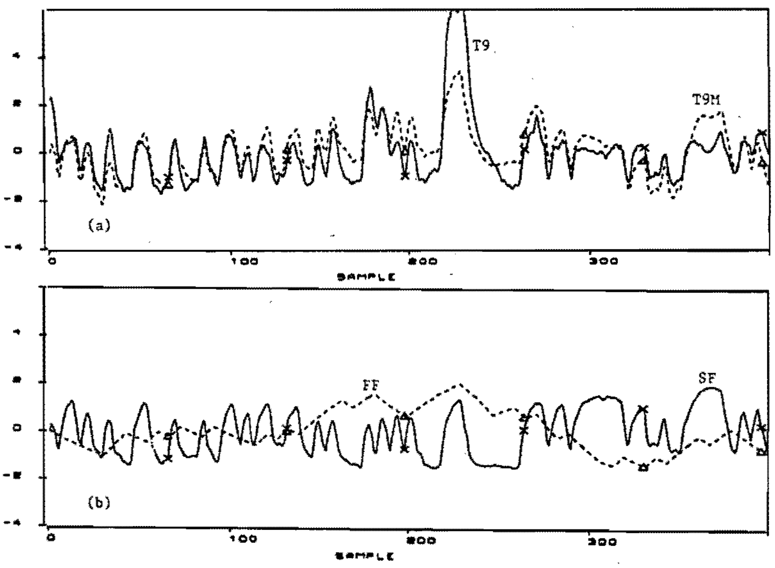


Figure 6.4.8:
(a) Model fit for temperature T9 "——" and model output T9M "— — —" on the identification interval
(b) individual contributions of the inputs FF and SF to the output T9M.

CHAPTER 7 CONCLUSIONS

In this thesis, I have focused attention on experimental modelling of continuous production processes in the process industry.
For this purpose, a new and comprehensive scheme for experimental modelling in industrial practice is described.
In contrast to conventional identification approaches, this scheme is based on an *interactive learning strategy* for process analyses and control syntheses in *real-time*.
A key property of the scheme is that the user may carry out a variety of analyses during the experiment, accumulating knowledge about the process and improving the experimentation accordingly.

The real-time approach lets the user:

- monitor the experiment;
- analyse the data in real-time, using signal analyses and identification, and inspect the intermediate results to check whether they contain the necessary information;
- use these results to instantly adapt the experiment, improving the information content of the data and/or switching experiments if insufficient (or unexpected) information is obtained;
- run a model in parallel to and synchronised with the process, predicting its future outputs and comparing them to the actually observed outputs.
- immediately test the actual performance of a designed control system. Many of the techniques used in modelling the process may also be used for testing and validating the control system.

The developed strategy comprises the following steps:

1. Definition of the goal and the purpose of the models
2. Investigation of the available process knowledge
3. Installation of equipment (to connect to the process)
4. Experiment control and data acquisition
5. Data conditioning and signal analyses
6. Identification
7. Model validation
8. Control system specification and design
9. Control system testing (and performance analysis)
10. Control system implementation and evaluation

PRIMAL (Package for Real-Time Interactive Modelling, Analyses and Learning), which is the main product of the work presented here, has been designed and constructed to implement this experimental modelling strategy. It was set up as a professional tool for real-time experimenting, analyses and control system design in industrial environments, particularly in the process industry.

Experience with PRIMAL has been built up in a variety of practical applications. Its most important properties proved to be:

- the facilities for coupling to industrial processes, and for designing and controlling experiments;
- the ability to carry out real-time analyses;
- the facilities for inspecting and processing intermediate results;
- the ability to adjust the experiment and the parameters of the active analyses;
- the availability of a complete set of compatible tools for all steps in the experimental modelling scheme;
- the ability to carry out analyses to real-time data, as well as to historical data, or even to both simultaneously;
- the automatic reporting;
- the storage of all results in a structured database and the easy retrieval and reconstruction of results;
- the powerful tools for manipulating real-time graphical presentations of the data;
- the open environment, readily extensible with new modules and conveniently interfaceable with other software packages.

Rather than adopting one single identification or control system design method, a variety of methods are made available. The user may select any appropriate method or, following the learning strategy, he may apply several methods in parallel and/or in series, and select the best results.

In this thesis, three cases are presented where PRIMAL has been applied in industry. The results proved that the experimental modelling strategy (and its methods), could indeed be applied successfully during normal process operation and did lead to improved insight and control of the process.
The results in applications not presented here, support this conclusion.

From practical experience with the experimental modelling scheme it follows that:

- Real-time analysis, interpretation of the intermediate results, and modifying the experiment accordingly, is often essential to arrive at good models.
- The data-conditioning step may be of decisive importance to identification.
- Preference should be given to identification methods which are robust and which give reliable results in the initial phases of the experiments, when little and possibly poor data is available.
  For this purpose, new low-order output error methods have been developed, using Monte-Carlo search combined with Least Squares estimation and zero-order search, providing initial estimates for an output error method. These methods prove to be valuable tools for estimating the delays and getting initial estimates of the process dynamics.

- Of the investigated identification methods:
  a) one-step ahead prediction error methods (PEM's), using an equation error or output error type of criterion,
  b) instrumental variable techniques,
  c) Monte-Carlo / Least-Squares (MCLS) techniques and zero-order search, optionally followed by an output error PEM,
  d) impulse response estimation and subsequent realisation,

  the output error PEM, using a MCLS start-up procedure, generally performed best, achieving the lowest output error in validation and cross validation.
- Equation error methods generally performed less than output error methods. However, in the cases presented in Chapter 6, the results were close to those of the output error methods.
- The four-step instrumental variable method often showed only minor improvements or even instability in the third step.
- Impulse response estimation failed in the case of short datasets and/or bad signal-to-noise ratio's. However, when successfull, it provided valuable information about the delays and the required model orders.
- Although the MCLS method was developed for problematic data, because of its robustness and ability to cope with unknown delays, high noise levels, and short datasets, its performance proved to be comparable to the other methods in the identification experiments also.

A striking finding is that experimental modelling, even with black box techniques, may yield valuable physical process insight.

Prospects

Up till now, the PRIMAL project has focused mainly on experimental modelling, and to a lesser extent on control system design. An important step in the future will be to fully exploit the PRIMAL-facilities to immediately test and analyse the actual in-plant performance of an improved control system.

Another domain of interest is the extension of the set of tools to gray/white models, i.e. incorporating physical process knowledge.

The advent of powerful computer facilities coupled to the plants in the process industry has provided new opportunities for improving plant performance. Given the availability of satisfactory sensors, no longer the problem is *how* to get data from the process, but *what* to do with it. The practical experience with PRIMAL and its methods have shown that experimental modelling may be exploited successfully to improve process control, thus bridging the gap between control theory and industrial practice.

PRIMAL, and the experience built up by its practical use, further serve us as a prototype for the design of the next generation of *real-time* tools for industrial processes. Extrapolating the PRIMAL strategy, the tools could be used in a flexible interactive environment, permanently coupled to a plant, for (expert assisted) model-based process monitoring, control and real-time optimisation.

APPENDIX A

Definition A.1 Quasi stationary signals

A signal $\{s(t)\}$ is quasi-stationary if

1) $\forall_{t\in\mathbb{Z}^+}\ Es(t) = m(t) \qquad \| m(t) \| \le C_1 \qquad C_1 \in \mathbb{R}$

2) $\forall_{t,r\in\mathbb{Z}^+}\ Es(t)s(r) = R(t,r) \qquad \| R(t,r) \| \le C_2 \qquad C_2 \in \mathbb{R}$

$$\lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} Es(t)s(t-\tau) \quad \text{exists for all } \tau \in \mathbb{Z} \qquad \square$$

The expectation operator is defined with respect to the stochastic components of s(t).

$$\text{Notation:}\quad \bar{E}s(t) := \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} Es(t) \tag{A.1}$$

The operator $\bar{E}$ may be applied to stochastic and deterministic signals. If applied to a stationary stochastic process:

$\bar{E}s(t) = Es(t)$ since the stochastic properties do not depend on t.

and if applied to a deterministic signal:

$$\bar{E}s(t) = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} s(t)$$

Definition A.2 Generalized weakly stationary processes

Let $\{x(t)\}\ t\in\mathbb{Z}$ be a n-dimensional discrete time stochastic process for which $Ex(t)$ and $Ex(t)x^T(s)$ exist for all $t,s \in \mathbb{Z}$. $\{x(t)\}\ t\in\mathbb{Z}$ is called generalized weakly stationary if:

$$R_x(t) := \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} Ex(t+\tau)x^T(t) \quad \text{exists for all } \tau\in\mathbb{Z} \qquad \square$$

Theorem A.3 Single realisation behaviour

Let $\{C_\theta(q) \mid \theta \in \mathcal{D}m\}$ be a uniformly stable family of filters, and assume that the family of deterministic signals $\{w_\theta(t)\}$, $\theta \in \mathcal{D}m$, is subject to $\forall\theta\ \forall t\ |w_\theta(t)| \leq C_w \qquad C_w \in \mathbb{R}$.
Define $s_\theta(t) := C_\theta(q)v(t) + w_\theta(t)$ for all $\theta \in \mathcal{D}m$.
Let $\{v(t)\}$ be subject to $v(t) = H_t(q)e(t)$ where $H_t(q)$ is a uniformly stable family of filters and $\{e(t)\}$ is a sequence of independent zero-mean random variables with $Ee(t)e^T(t) = \Lambda_t$ and bounded fourth moments.
Then:

$$\sup_{\theta\in\mathcal{D}m} \left\| \frac{1}{N} \sum_{t=1}^{N} [s_\theta(t)s_\theta^T(t) - Es_\theta(t)s_\theta^T(t)] \right\|_2 \longrightarrow 0 \qquad \text{(A.2)}$$

w.p. 1 as $N \longrightarrow \infty$ □

Proof: [Ljung, 1987], Theorem 2B.1.

Definition A.4 Model set

A model set is defined as a collection of models:

$$\mathcal{M} = \{\mathcal{M}_\alpha(q) \mid \alpha \in \mathcal{D}\}$$

where the index α covers an index set $\mathcal{D}$. □

Remark:
In the context of Ljung [1987] the models constituting the model set are predictor models (i.e. residual generating models).

Definition A.5 Model parametrisation

A model parametrisation of the elements in the model set $\mathcal{M}$ is a mapping $P: \mathcal{D}m \rightarrow \mathcal{M}$ for some parameter set $\mathcal{D}m$. □

Let $\mathcal{M}$ be a model set parametrised by $\theta \in \mathcal{D}m$. This may be written as: $\mathcal{M} = \{\mathcal{M}(\theta) \mid \theta \in \mathcal{D}m\}$

APPENDIX B

The following list gives an overview of currently available PRIMAL modules. A full description of each module may be found in the PRIMAL manuals [Van der Linden & Renes, 1989 a-e]

IDM	Interface definition module	
EDM	Experiment definition module	
ECM	Experiment control module	
IMPORT	Modules for importing files from other packages	
EXPORT	Modules for exporting files to other packages	
MATLAB	Interface to MATLAB	
PLOT	Graphical presentation module	
LIST	Presentation of dataset contents on screen	
PRINT	Presentation of dataset contents on a printer	
PREFILTER	Recursive data conditioning	
FILTER	Data conditioning	
FFT	Discrete Fourier transform	
RFT	Reverse discrete Fourier transform	
CORRELATOR	Recursive correlation analysis	
SPECTRUM	Spectral analysis	
RPE	Recursive prediction error method	(SISO)
RPEM	Recursive prediction error method	(MISO)
EMM	Recursive extended matrix method	(SISO)
MARKOV	Recursive impulse response estimation	(MIMO)
TRANSFER	Direct transfer function estimation	(SISO)
DLS	Least squares method	(MIMO)
CUIDORZI	Guidorzi's method	(MIMO)
GIV	Bootstrap-IV + delay grid search	(MISO)
MCR	Monte-Carlo search + zero order opt.	(MIMO)
MCRPEM	Monte-Carlo search + output error PEM	(MISO)
IVM	Four-step Instrumental variables	(MIMO)
MFDSS	Matrix fraction model to state space conversion	
SSMFD	State-space to Matrix Fraction model conversion	
SIMSYS	System simulation	
MODELTST	Recursive model simulation + process monitoring	
MTEST	Model validation module + residual analysis	
RESPONSE	Analysis of subsystem contribution	
MATEDIT	Model editor	
RSM	Dummy process simulator	

ORDERTEST	Equation error dimension test
HANKDIM	Order test based on the Hankel matrix
HANKEL	Realisation method based on the Hankel matrix
KALMAN	Kalman filter
LQG	LQG control design
PPLCR	Robust pole placement

REFERENCES

Anderson, B.D.O. and J.B. Moore (1979)
Optimal Filtering
Prentice-Hall, Englewood Cliffs, N.J.

Åström, K.J. (1985)
Process control - Past, present and future.
IEEE Control Systems Magazine, August 1985, pp. 3-10

Åström, K.J. and B. Wittenmark (1984)
Computer controlled systems: theory and design.
Prentice-Hall, Englewood Cliffs, N.J.

Backx, A.C.P.M. (1987)
Identification of an industrial process: a Markov parameter approach.
Ph-D. Thesis, Eindhoven University of Technology, 1987.

Backx, A.C.P.M. and A. Damen (1989)
Identification of industrial MIMO processes for fixed controllers.
Journal A, Vol. 30, no 1, 1989

Bartman R.V. (1981)
Dynamically reconciled control
Chemical Engineering Progress, September 1981

Baur, U. (1976)
On-line Parameterschätzverfahren zur Identifikation linearer, dynamischer Prozesse mit Prozessrechnern.
Dissertation Universität Stuttgart, Kernforschungszentrum Karlsruhe

Berben, P.J. (1987)
Een optimale IV-parameterschattingsmethode voor PRIMAL.
M.Sc. thesis, Eindhoven University of Technology, 1987 (in Dutch)

Betlem B. and O. Rademaker (1979)
Internal report NR-469, Systems and Control Group, Dep. of Physics, Eindhoven University of Technology, 1979.

Bohlin, T. (1987)
Encyclopedia of Systems and Control (M.Singh, ed.)
Pergamon Press, Oxford.

Box, M.J., D. Davies and W.H. Swan (1969)
Non-linear Optimization Techniques
Oliver and Boyd, Edinburgh, 1969.

Cauwenberghe, A.R. van (1985)
Trends in automatic control applications.
Journal A, Vol. 26, no 1, January 1985

Cellier, F.E. and C.M. Rimvall, (1986)
Computer-aided control systems design: techniques and tools.
CERL-report: 86/04. University of Arizona.

Damen, A.A.H. and A.K. Hajdasinski (1982)
Practical tests with different approximate realization based on the singular value decomposition of the Hankel matrix.
Proc. 6th IFAC Symposium on identification and system parameter estimation, Washington

Davies, W.D.T. (1970)
System identification for self-adaptive control.
Wiley-Interscience, New York

Dennis, J.E. Jr and R.B. Schnabel (1986)
Numerical methods for unconstrained optimization and nonlinear equations.
Prentice-Hall series in computational mathematics.

Eykhoff, P. (1974)
System identification - Parameter and state estimation.
Wiley and Sons, London

Eykhoff, P. Ed. (1981)
Trends and progress in system identification.
Pergamon Press, IFAC series for graduates, researchers & practising engineers, Vol. 1, 1981

Freeman, T.G. (1985)
Selecting the best linear transfer function model.
Automatica, Vol. 21, no 4, pp. 361-370, 1985

Gerlings, P.H.M. (1987)
The Hankel realization method in PRIMAL.
M.Sc. Thesis, Eindhoven University of Technology, 1987

Gevers, M.R. (1986)
ARMA models, their Kronecker indices and their Mc Millan degree.
Int. J.Control, Vol. 43, no 6, 1986, pp. 1745-1761

Gill, P. and W. Murray (1981)
Practical optimization.
Academic Press, London.

Godfrey, K.R. (1969)
The theory of the correlation method of dynamic analysis and its application to industrial processes and nuclear power plant.
Measurement and Control, Vol. 2, May 1969, pp. 65-72.

Goodwin, G.C. and R.L. Payne (1977)
Dynamic system identification: experiment design and data analysis.
Academic Press, New York

Guidorzi, R.P. (1975)
Canonical structures in the identification of mulitvariable systems.
Automatica, Vol. 11, pp. 361-374, 1975

Hannan, E.J. and M. Deistler (1988)
The statistical theory of linear systems.
John Wiley and sons, New York.

Hof, P.M.J. van den, and P.H.M. Janssen (1985)
Some Asymptotic Properties of Multivariable Models identified by Equation Error Techniques.
EUT Report 85-E-153, Eindhoven University of Technology.

Hogendoorn, M.J. (1988)
Modellering van een kristallisatie-proces.
M. Sc. thesis, Eindhoven, University of Technology (in dutch)

Isermann, R. (1974)
Prozessidentifikation.
Springer-Verlag, Berlin

Isermann, R. (1988)
Identifikation dynamischer Systeme.
Springer-Verlag, Berlin

Jackson, L.B. (1986)
Digital filters and signal processing
Kluwer ac. publ., The Netherlands.

Jakeman, A.J. and P.C. Young (1979)
Refined instrumental variable methods of recursive time-series analysis. Part II: Multivariable systems.
International Journal of Control, Vol. 29, pp. 621-644.

Janssen, P.A. (1987)
Practical aspects of process identification with PRIMAL.
MSc. Thesis, Eindhoven University of Technology, 1987.

Janssen, P.H.M. (1988)
On model parametrization and model structure selection for identification of MIMO-systems.
Ph-D Thesis, Eindhoven University of Technology, 1988.

Kailath, T. (1980)
Linear systems.
Prentice-Hall, Englewood Cliffs, N.J.

Kung, S. (1979)
A new identification and model reduction algorithm via singular value decompositions.
Proc. 12th Annual Asilomar Conf. Circ. Syst. Comp. pp. 705-714.

Lanen, M.M.M. van, (1988)
Grafische faciliteiten voor het gebruik van PRIMAL op een VAX-werkstation.
MSc. Thesis, Eindhoven University of Technology, 1988 (in Dutch).

Linden, R.J.P. van der, W.A. Renes and P.A. Janssen, (1987)
Interactive on-line signal analysis and identification,
Proc. IASTED conf. on identification, modelling and simulation, Paris, 1987.

Linden, R.J.P. van der, W.A. Renes and O. Rademaker (1988),
PRIMAL: a Package for Real-time Interactive Modelling, Analysis and Learning.
Journal A, Vol. 29, no. 1, March 1988, pp 10-16.

Linden, R.J.P. van der and W.A. Renes (1989a)
PRIMAL User Manual.
ITP-TUE/TNO report.

Linden, R.J.P. van der and W.A. Renes (1989b)
PRIMAL Command Language Reference
ITP-TUE/TNO report.

Linden, R.J.P. van der and W.A. Renes (1989c)
PRIMAL Module Reference Manual
ITP-TUE/TNO report.

Linden, R.J.P. van der and W.A. Renes (1989d)
PRIMAL Process Interface Manual
ITP-TUE/TNO report.

Linden, R.J.P. van der and W.A. Renes (1989e)
PRIMAL Module Development Manual
ITP-TUE/TNO report.

Ljung, L. (1976)
On the consistency of prediction error identification methods.
In R.K. Mehra and D.G. Lainiotis eds.: System Identification - Advances and Case studies.
Academic Press, New York.

Ljung, L. (1977)
Analysis of recursive stochastic algorithms.
IEEE Trans. on Autom. Control, vol AC-22, pp. 551-575.

Ljung, L. (1983)
Theory and practice of recursive identification.
The MIT Press, Cambridge, Massachusetts.

Ljung, L. (1987)
System Identification - Theory for the user.
Prentice-Hall, Englewood Cliffs, N.J.

Ljung, L. Ed. (1988)
Control Theory 1984-1986.
Automatica, Vol. 24, No. 4. pp. 573-583, 1988.

Mehra, R.K. (1974)
Optimal input signals for parameter estimation in dynamic systems
- a survey and new results.
IEEE Trans. Automatic Control, Vol. AC-19, pp. 753-768

Meyer, B. (1987)
Object-oriented software construction.
Prentice Hall, New York.

Rademaker, O., J.E. Rijnsdorp and A. Maarleveld (1975)
Dynamics and control of continuous distillation units.
Elsevier Scientific Publ. Comp., Amsterdam.

Renes, W.A. (1984)
PRIMAL - Programmapakket voor real-time interactieve procesanalyse.
M.Sc. Thesis, Eindhoven University of Technology, 1984.

Renes, W.A. and R.J.P. van der Linden (1987)
Data acquisition for modelling and control in an industrial environment.
Proc. IASTED conf. on Identification, modelling and simulation, Paris, 1987.

Söderström, T. (1989)
System Identification.
Prentice Hall International, Hemel Hempstead.

Söderström, T. and P. Stoica (1983a)
Instrumental variable methods for system identification.
Springer Verlag, Berlin.

Söderström, T. and P. Stoica (1983b)
Optimal instrumental variable estimation and approximate implementations.
IEEE Trans. on Autom. Control, Vol. AC-28, July 1983, No. 7, pp. 757-772.

Stanley, W.D. (1975)
Digital signal processing.
Mc.Craw-Hill, New York.

Stoica, P.C. et. al. (1984)
On the asymptotic accuracy of pseudo-linear regression algoritms.
Int. J. of Control, Vol. 39, no. 1., pp. 115-126

Stoica, P. et. al. (1986)
Model structure selection by cross validation.
Int. J. Control, Vol 43, pp. 1841-1878.

Tolboom, W.J. (1989)
De modellering en het regelaarontwerp van een Tolueen destillatie toren.
MSc. Thesis, Eindhoven University of Technology, 1989 (in Dutch)

Tulleken, H.J.A.F. (1988)
A generalised binary noise test-signal concept for improved identification-experiment design.
Preprints 8th IFAC/IFORS Symposium on Identification and System parameter estimation, Beijing, pp. 847-853

Vucht, G.N.M. van (1987)
Identification of a glass-feeder process.
MSc. Thesis, Eindhoven University of Technology.

Wahlberg, B. and L. Ljung (1986)
Design variables for bias distribution in transfer function estimation.
IEEE Trans. Automatic Control, Vol. AC-31, pp. 134-144

Wirth, N. (1976)
Algorithms + Data structures = Programs
Prentice Hall, Englewood Cliffs, NY.

Wolovich, W.A. (1974)
Linear multivariable systems.
Applied mathematical sciences, Vol. 11, Springer Verlag New York.

Young, P.C. (1984)
Recursive estimation and time series analysis.
Springer, New York.

Zeiger, H.P. and A.J. McEwen (1974)
Approximate linear realizations of given dimension via Ho's algorithm.
IEEE Trans. on Automatic Control, Vol. AC-19, pp. 153-167.

NOTATIONS, SYMBOLS AND ABBREVIATIONS

Mathematical notations:

arg min	: minimising argument
$AsN(m,P)$	: asymptotically normal distribution with mean m and covariance matrix P
col(A)	: vector containing the columns of matrix A
deg	: degree
det	: determinant
min	: minimise
sol	: solution of the equation
tr	: trace
A^T	: transpose of A
A^{-1}	: inverse of A
$\sim$	: equivalence relation
∞	: infinity
$\Vert . \Vert$	: norm of a vector or matrix
$V'(\theta)$	: derivative of V w.r.t. θ

Symbols:

$e(t)$	: disturbance at time t
$G(q^{-1};\theta)$	: process transfer operator
$G_i(\theta)$	: the coefficient matrix of q^{-i} in $G(q^{-1};\theta)$
$h(t)$	: impulse response
$H(q^{-1};\theta)$	: noise transfer operator
$M(k)$	: k'th Markov parameter
na,nb,nc,nd,nf	: degrees of $A(q^{-1};\theta),\ldots,F(q^{-1};\theta)$
$n\theta$	: dimension of θ
N	: number of samples
p	: number of inputs
P	: parameter covariance matrix
q	: number of outputs
$R_u(\tau)$	: correlation function of u(t)
$R_{uy}(\tau)$	: cross-correlation function between u(t) and y(t)
t	: discrete time
T_o	: sampling interval
$u(t)$	: input signal at time t
$V_N(\theta,Z^N)$	: criterion function
$x(t)$	: state vector at time t
$y(t)$	: output signal at time t
$\hat{y}(t\|t-1)$	: one step ahead prediction of the output
$ym(t)$	: model output
$Z(t)$	: instrument matrix

Z^t	: vector of delayed inputs and outputs at time t
α	: step length, amplitude
$\mathbb{C}$	: the complex numbers
Λ	: noise covariance matrix
$\delta_{i,j}$	: Kronecker delta
$\delta(t)$	: unit pulse
$\mathcal{D}m$	: domain of θ
$\varepsilon(t)$	: prediction error
E	: expectation operator
$\bar{E}$	: generalised expectation operator
$\varphi(t)$	: regression vector
$\Phi_y(\omega)$	: spectrum of y(t)
$\Phi_{uy}(t)$	: cross spectrum of u(t) and y(t)
I	: identity matrix
I_p	: pxp identity matrix
λ	: forgetting factor, minimal PRBS clock period
$\mathcal{M}$	: model set
$\eta(t)$	: vector of instruments at time t
$\psi(t,\theta)$	: gradient of $-\varepsilon(t)$ w.r.t. θ
Ω	: weighting matrix
q^{-1}	: backward shift operator
$\mathbb{R}$	: the real numbers
σ	: standard deviation
τ	: time lag
θ	: parameter vector
θ_o	: true parameter vector
$\hat{\theta}$	: parameter estimate
ω	: angular frequency
$\mathbb{Z}$	: the integers { .. -2, -1, 0, 1, 2, ...}
$\mathbb{Z}^+$	: the nonnegative integers {0, 1, ...}

Abbreviations:

ADC	: Analog to Digital Conversion
AIC	: Akaike's Information theoretic Criterion
AR	: Auto Regressive
ARIMA	: Auto Regressive Integrating Moving Average
ARMA	: Auto Regressive Moving Average
ARMAX	: Auto Regressive Moving Average eXogeneous
ARX	: Auto Regressive with eXogeneous inputs
DAC	: Digital to Analog Conversion
DCS	: Distributed Control System
DCM	: Data Generating Model
DLS	: PRIMAL Module (See Appendix B)
ECM	: Experiment Control Module
ee	: equation error method
EM-scheme	: Experimental Modelling scheme
FIR	: Finite Impulse Response
FPE	: Final Prediction Error criterion

GIV	:	PRIMAL Module (see Appendix B)
GUIDORZI	:	PRIMAL Module (see Appendix B)
HANKEL	:	PRIMAL Module (see Appendix B)
IDM	:	Interface Definition Module
IV	:	Instrumental Variables
IVM	:	Instrumental Variable Method (see Appendix B)
LS	:	Least Squares
MA	:	Moving Average
MARKOV	:	PRIMAL Module (see Appendix B)
MCR	:	PRIMAL Module (see Appendix B)
MCRPEM	:	PRIMAL Module (see Appendix B)
MFD	:	Matrix Fraction Description
MIMO	:	Multi Input Multi Output
MISO	:	Multi Input Single Output
mre	:	mean relative output error
oe	:	output error method
PCS	:	Process Control System
PEM	:	Prediction Error Method
pdf	:	probability density function
PIM	:	Process Interface Module
PEM	:	Prediction Error Method
PLR	:	Pseudo-Linear Regression
PRBS	:	Pseudo Random Binary Signal
re	:	relative output error
ree	:	recursive equation error method
roe	:	recursive output error method
RCM	:	Residual Generating Model
RPEM	:	PRIMAL Module (see Appendix B)
SIMO	:	Single Input Multi Output
SISO	:	Single Input Single Output
SVD	:	Singular Value Decomposition
wp1	:	with probability one

SAMENVATTING

In de beheersing van continue produktieprocessen speelt het dynamische procesgedrag vaak een belangrijke rol, zodat inzicht in en kennis van de dynamica, bij voorkeur in de vorm van een voldoende betrouwbaar model, is vereist. Deze kennis kan langs theoretische en experimentele weg worden opgebouwd.
In de experimentele modelvorming wordt aan de hand van metingen van ingangs- en uitgangssignalen een mathematisch model van het proces gevormd, dat vervolgens gebruikt kan worden voor onder andere diagnose, procesbewaking, voorspelling van procesgedrag en automatische regeling.

Dit proefschrift beschrijft een strategie voor experimentele modelvorming, de ontwikkeling van een pakket voor het ondersteunen van deze strategie, en de toepassing en evaluatie van de strategie (en zijn methoden) op industriële produktieprocessen.

De ontwikkelde strategie wordt gekenmerkt door *interactief leren* en *real-time analyse*. Het omvat de definitie van projectdoelen, onderzoek van proceskennis, installatie van apparatuur, experimentontwerp, uitvoering van metingen, dataconditionering, signaalanalyse, identificatie en modelvalidatie, als ook ontwerp, beproeving en evaluatie van regelaars.

In tegenstelling tot de gangbare "off-line" aanpak van modelvorming, staat het belang van real-time analyse centraal in deze strategie. Het plannen en uitvoeren van de metingen zijn hierin essentiële onderdelen.
De analyse van de gemeten data kan interactief, in real-time, uitgevoerd worden, waarbij de gebruiker kennis van het procesgedrag opbouwt en vervolgens deze kennis gebruikt om het experiment en de analyses te verbeteren.
Ook kunnen de gevormde modellen synchroon aan het proces meelopen als "voorspellers", waarvan de uitkomsten direct vergeleken kunnen worden met het werkelijke procesgedrag. Tenslotte kunnen de ontworpen regelsystemen gebruikt worden voor simulaties, die (geleidelijk) in toepassing worden gebracht onder voortdurende controle van hun prestaties, waarbij de daarbij verzamelde kennis gebruikt kan worden om hun werking verder te verbeteren. Aldus kan in alle projectfasen effectief geleerd worden én snel en doeltreffend van het geleerde geprofiteerd worden.

Om de strategie voor experimentele modelvorming in de praktijk te ondersteunen is het pakket **PRIMAL** (Package for Real-time Modelling, Analyses and Learning) ontworpen en ontwikkeld. Het biedt de mogelijkheden om aan industriële processen te koppelen, metingen te verrichten, in real-time de gemeten data te analyseren, modellen te schatten en het ontwerp en de beproeving van regelsystemen direct uit te voeren.

Voor alle stappen in het experimentele modelvormingsschema zijn in PRIMAL diverse methoden ontwikkeld en geïmplementeerd. Speciale aandacht is hierbij geschonken aan de identificatie stap, waaronder prediction-error methoden, instrumentele variabelen methoden en hier beschreven nieuw ontwikkelde varianten op output-error methoden. Aan de hand van de praktijktoepassingen is een evaluatie van zowel PRIMAL, als van de prestaties van deze methoden uitgevoerd.
Daarnaast is aandacht geschonken aan de bewerking (conditionering) van de ruwe procesdata, die in de praktijk vaak van meer belang blijkt te zijn dan de keuze van de identificatiemethode.

PRIMAL is reeds op een aantal industriële processen toegepast, waarvan de resultaten op een glasproductieproces, een para-xyleen kristallisatie-proces en een tolueen-xyleen destillatietoren in dit proefschrift worden beschreven.
De resultaten tonen aan, dat experimentele modelvorming van industriële processen, met behulp van moderne multivariabele methoden, succesvol toegepast kan worden en tot waardevolle fysische inzichten in het proces en de regeling ervan kan leiden.

CURRICULUM VITAE

19 april 1958	Geboren te Geldrop
1970 - 1976	Atheneum-b aan het Augustinianum te Eindhoven
1976 - 1983	Studie Technische Natuurkunde, Technische Universiteit Eindhoven.
1983 - 1985	Wetenschappelijk assistent bij de vakgroep Systeem- en Regeltechniek van de faculteit Technische Natuurkunde, Technische Universiteit Eindhoven.
1985 - 1989	Universitair Docent bij de vakgroep Systeem- en Regeltechniek van de faculteit Technische Natuurkunde, Technische Universiteit Eindhoven.
vanaf 1 januari 1990	Werkzaam bij het Instituut Informatie-Technologie voor Produktieautomatisering ITP/TUE-TNO te Eindhoven.

STELLINGEN

behorende bij het proefschrift

"Design and Application of PRIMAL:

A Package for Experimental Modelling of Industrial Processes"

van

R.J.P. van der Linden

Eindhoven

Mei 1990

- I -

Experimentele modelvorming van produktieprocessen in de procesindustrie dient bij voorkeur gekoppeld aan het proces en in **real-time** plaats te vinden (dit proefschrift; Ljung (1983) Theory and Practice of Recursive Identification; Isermann (1988) Identifikation Dynamischer Systeme).

- II -

Voor een brede toepassing van experimentele modelvorming in de praktijk is een "software environment" vereist, die niet alleen geschikte analysemethoden bevat, maar tevens het leerproces van de gebruiker ondersteunt door krachtige real-time interactieve faciliteiten, rapportage en gestructureerde gegevensopslag (Söderström & Stoica (1989): System Identification, en dit proefschrift).

- III -

Identificatie op basis van black-box technieken kan belangrijk fysisch inzicht in een proces opleveren, waarmee door aanpassingen in het proces soms meer bereikt kan worden dan door op basis van het model ontworpen regelingen (dit proefschrift).

- IV -

De praktische aspecten van identificatie worden in de literatuur onvolledig belicht: niet vanwege het geringe belang, maar wegens het ontbreken van een sluitende aanpak (dit proefschrift, zie bijvoorbeeld voor wat data conditionering betreft: Ljung (1987) System Identification - Theory for the user; Backx (1987) Identification of an Industrial Process - A Markov Parameter Approach).

- V -

Over de toepassing van kleine testsignalen op industriële processen tijdens normaal bedrijf wordt vaak te moeilijk gedaan, terwijl dit in praktijk zelden of nooit tot problemen leidt.

- vi -

Zolang in het veld van de experimentele modelvorming de wijze waarop een experimentator kwalitatief grafische presentaties interpreteert niet goed te expliciteren is, mag van expert-systemen niet veel verwacht worden en blijft de experimentator een onmisbare schakel.

- vii -

De procesindustrie heeft belang bij procesontwerpers die rekening kunnen en durven houden met de dynamica.

- viii -

Aangezien in steeds meer wetenschappelijk onderzoek de ontwikkeling van programmatuur een belangrijke plaats inneemt, dient deze professioneel en voor hergebruik ontworpen te worden. De universiteit zou hieraan in de opleiding en in de uitvoering van het onderzoek veel meer aandacht moeten schenken.

- ix -

In een omvangrijk universitair software project als PRIMAL, is de standaardisatieregel dat alles wat niet aan de standaard voldoet wordt weggegooid, de beste manier om studenten te bewegen om daadwerkelijk volgens de standaard tewerk te gaan.

- x -

De colleges een half uur eerder laten beginnen heeft op korte termijn waarschijnlijk nauwelijks gevolgen voor de tijden waarop studenten met een OV-jaarkaart reizen.

- xi -

Een belangrijke oorzaak voor de matige telefonische bereikbaarheid van wetenschappers aan deze universiteit is, dat zij niet goed weten om te gaan met de mogelijkheden die het telefoonsysteem biedt (de TUE telefoongids blz. 28-38).