Trajectory-Based Model Reduction of Nonlinear Systems

Öhman, Martin

1998

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

Trajectory-Based Model Reduction of Nonlinear Systems

Martin Öhman

Department of Automatic Control
Lund Institute of Technology

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

Document name
LICENTIATE THESIS

Date of issue
October 1998

Document Number
ISRN LUTFD2/TFRT--3223--SE

Author(s)
Martin Öhman

Supervisor
Anders Rantzer

Sponsoring organisation
NUTEK and Sydkraft Research Foundation.

Title and subtitle
Trajectory-Based Model Reduction of Nonlinear Systems

Abstract

The first part of this thesis concerns model reduction of nonlinear models. Physical insight in a model can be used to indicate which parts of the model that can be reduced. Based on linearization around specific trajectories, a systematic procedure is suggested to estimate the error caused by reduction. Optimization is used to adjust parameters in a reduced model to minimize the error. This is applied to two examples.

The first example is an inverted pendulum connected to a rotating beam. It is shown that, for a swing-up trajectory the pendulum can be quite well approximated by a pendulum on a cart. The second example is a drum boiler, modeled with a second order as well as a fourth order model. For a experimental trajectory it is possible to reduce large parts of the model, adjust the remaining parameters and get a model that is almost as correct as the original one.

The procedures discussed above to estimate error involve approximations. Ideas are also presented in this thesis about how to find a strict upper bound for the error. The Small Gain Theorem is used on the Taylor expansion of the error.

The second part of the thesis contains the paper *Implementation aspects of the PLC standard IEC 1131-3.* IEC 1131-3 is a standard for PLCs defining four programming languages and a type of Grafcet, Sequential Function Charts (SFC). An object oriented prototype of SFC and the language Function Block Diagram has been implemented. Various execution methods are discussed. Algorithms for local and global sorting are implemented and evaluated. The standard is found to be unclear in some parts.

Key words
Nonlinear, Model Reduction, Linearization, Trajectory, Pendulum, Boiler, Error Bounds. Grafcet, IEC 1131, Programmable logic controller, Standard.

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
110

Recipient's notes

Security classification

Trajectory-Based Model Reduction of Nonlinear Systems

Martin Öhman

Department of Automatic Control
Lund Institute of Technology
Lund, October 1998

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

ISSN 0280–5316
ISRN LUTFD2/TFRT–3223–SE

Printed in Sweden by KFS AB.
Lund 1998

Contents

Preface

Research in automatic control covers many different areas. Solving a control problem from scratch typically involve e.g.,

- Modeling and identification
- Analysis
- Synthesis
- Implementation

This thesis contains two parts and contributes to the research within two quite different parts of automatic control research, the first and fourth point above. After the modeling and identification process, one often end up with a too complicated model. Then model reduction is needed. Many models are highly nonlinear and it is interesting to be able to keep important nonlinearities while doing model reduction. The Part I of the thesis contributes to the research about nonlinear model reduction along specific trajectories.

A process model can then be used for analysis and synthesis. The synthesized controller needs to be implemented. Many industrial controllers are implemented in PLC, Programmable Logical Computers. PLCs are programmed in many different languages, often not compatible. An attempt to standardize the PLC programming is the standard IEC 1131-3. The Part II of this thesis contains a paper describing a prototype implementation of the standard. The title is "Implementation aspects of the PLC standard IEC 1131-3" and it will be published in Control Engineering Practice.

Acknowledgments

This work has been supported by the Swedish National Board for Industrial and Technical Development (NUTEK) and by Sydkraft Research Foundation.

I would like to thank my supervisors Sven Erik Mattsson, Karl Johan Åström and Anders Rantzer. Special thanks to Anders for always having time and being optimistic and inspiring. I have enjoyed our discussions. The second part of the thesis is based on a project done together with Stefan Johansson under supervision of Karl-Erik Årzén. Thanks to both of them for a valuable cooperation. Finally thanks to my friend Lennart Andersson for guiding and supporting me during my time at the departement.

Part I

1

Introduction

1.1 Background

This section discusses why it is relevant to do research in nonlinear model reduction. Model reduction in control can be divided into

- Reduction of a process model
- Reduction of a controller model
- Reduction of a closed loop model

The model reduction methods presented in this thesis are mainly intended for reduction of process models. Reduction of process models are interesting for several reasons:

- A reduced model is often easier to analyze and understand.
- A model is often used for simulations and the simpler the model is, the easier it is to implement and maintain.
- The controller design process is easier with a simpler process model.
- A reduced process model often leads to simpler controller which needs less computational power when implemented. This is the case e.g., with Feedback Linearization and Internal Model Control.

Most model reduction methods are applicable only to linear models. However, many process models derived from physical principles are nonlinear. If linear reduction methods are to be used the models need to be linearized, often with significant loss of accuracy. Therefore it is interesting to do model reduction of nonlinear models and this is the topic of this thesis.

1.2 Research approach

This thesis is dealing with reduction of non-linear models. It's generally hard to make simplifications that are valid everywhere. The approach is therefore to find a simplified model that is valid in a neighborhood of some typical trajectories.

Summary of Thesis ideas

This section summaries the ideas for model reduction of nonlinear models presented in the thesis. For details, see Chapter 2 and Chapter 4

In this thesis, models on the form $F(\dot{x}(t), x(t)) = 0$ are considered. Suppose that a model contains several parameters and can be written on form of Equation (2.2):

$$0 = F(\dot{x}(t), x(t), a_0, \dots, a_m) \qquad x(0) = x_0 \tag{1.1}$$

where $a_0 = a_1 = \ldots = a_m = 1$ is the nominal model. The user suggests which parameter to neglect and the formulation (1.1) is chosen so that setting $a_0 = 0$ and keeping $a_1 = a_2 = \ldots = a_m = 1$ gives the reduced model. A trajectory is specified by the choice of initial condition x_0. A possible input signal can be included in the function F. Based on a linearization around the trajectory a differential equation describing the error $\tilde{x}(t)$ is formulated in (2.12)

$$\dot{\tilde{x}}(t) \approx \left(\frac{\partial F}{\partial \dot{x}}\right)^{-1} \left(-\frac{\partial F}{\partial x}\tilde{x}(t) - \frac{\partial F}{\partial a_0}\right) \tag{1.2}$$

This equation approximates of the error in the states, caused by the truncation. A possible second step is to adjust the remaining parameters in the model with some small coefficient ε_i to minimize the error.

$$0 = F(\dot{\bar{x}}(t), \bar{x}(t), 0, 1 + \varepsilon_1, \dots, 1 + \varepsilon_m) \tag{1.3}$$

The goal is now to minimize the error in some sense, for example

$$\min_{\varepsilon} e(\varepsilon, T) = \min_{\varepsilon} \|\tilde{x}(\varepsilon, t)\| = \min_{\varepsilon} \int_0^T \sum_i \tilde{x}_i(\varepsilon, t)^2 dt \qquad (1.4)$$

This is done in this thesis by simulating the Jacobian $G(\varepsilon) = \partial f/\partial \varepsilon$ and Hessian $H(\varepsilon) = \partial^2 f/\partial \varepsilon^2$ and doing a line search in the Newton direction.

$$\varepsilon = H(0)^{-1} G(0)$$

Equation (1.2) only gives an approximation of the error. In Chapter 4 it is shown that in some cases an upper bound can be found using the small gain theorem. These calculations are only done for models in the explicit form

$$\dot{x}(t) = f(x(t), a_0, \ldots, a_m)$$

but could be extended to systems in the implicit form (1.1).

Examples

The ideas presented above are applied to two different examples.

The first example is a rotating inverted pendulum, shown in Figure 2.1. An inverted pendulum is connected to a rotating beam. The acceleration of the pivot point of the beam is considered as the control signal. The control goal is to swing up the pendulum from downward to upward equilibrium. It is shown that the pendulum, for the given swing trajectory, can be quite well approximated by a pendulum on a cart, shown in Figure 2.2.

The second example is a drum boiler model. Input signals used for identification of a boiler at Öresundsverket in Malmö are used. For this trajectory it is possible to reduce large parts of the model and get a model that is almost as correct as the original one. If the remaining parameters are adjusted the error is reduced significantly, but then some physical insight might get lost. A drum boiler is a classical control problem. It is interesting to find out that a reduced model can capture most of the behavior for a specific trajectory designed for identification purposes.

Contribution

This thesis contributes to current research in nonlinear model reduction. The main contributions are:

- New ideas for reduction of nonlinear models along specific trajectories.
- An estimation of the error in each state, caused by the reduction.
- In some cases, a strict upper bound on the error.

The ideas are developed for and tested on two rather different examples and it is likely that they could be useful also for other models and other trajectories.

The criterion for using the methods is that the user knows a trajectory or set of trajectories that is typical for what the model is intended to be used for and also have hypothesis about what parts of the model to eliminate.

Computer tools

Matlab, see Matlab (1996), has been used for simulations, since it has a nice and concise matrix notation, an intuitive and consistent user interface and extensive routines to present data graphically. The simulation facilities in Matlab are good enough for the small sized models used as examples in this thesis. For larger scale simulations it would be natural to use a simulation package like Modelica, see Eurosim (1998). Some Jacobians and Hessians are derived using Maple, see Maple (1992).

1.3 Related work

This section describes some research areas related to the work done in this thesis.

This thesis concerns reduction of models to simpler expressions but of the same order. Model reduction in general, however, mostly involve reduction to a system of lower order. This is here referred to as model order reduction.

	Nonlinear	Keep physical insight	Trajectory specific	Reduce model order
Balanced Realization	-	-	-	X
Trajectory-Specific Model Reduction by P. Mavrikis	-	?	X	X
Perturbation Method described by K. H. Khalil	X	X	-	-
Reduction of Chemical Systems by Linda Petzold	X	X	X	-
Nonlinear System Identification by Jonas Sjöberg et al	X	?	X	X
Reduction as discussed in this thesis	X	X	X	-

Figure 1.1 Similarities and differences between this thesis and related work.

There exist a large number of model reduction methods, see Andersson (1999) for an overview. Here only those methods most closely related to this thesis are reviewed. A schematic view of their properties is given in Figure 1.1.

Model order reduction of Linear systems using Balanced realization

Most model order reduction methods are designed for linear systems only. Model reduction of linear systems can be done in several different ways, e.g. with continued fraction approximation or via a balanced realization, both described in Johansson (1993).

Balanced realization is one of the most well known approaches to model order reduction of linear systems. The main purpose of doing balanced realization is to find a state space description that separates important states from non-important ones. The less important states can then be reduced.

Consider a system on the form

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

A state space transformation $z = Tx$ is applied. T is constructed via a Cholesky factorization to give realization where the reachability Gramian

$$P = \sum_{k=0}^{\infty} A^k B B^T (A^T)^k$$

and the observability Gramian

$$Q = \sum_{k=0}^{\infty} (A^T)^k C^T C A^k$$

are equal.

$$P = Q = diag(\sigma_1, \ldots, \sigma_n)$$

The system is then "equally reachable and controllable". Then the singular values can be used to do model order reduction. States corresponding to small singular values are neglected. Comparing the magnitude of the singular values $\sigma_i = \sqrt{\lambda_i(PQ)}$ indicates which states that can be eliminated. Small singular values correspond to less important states. The states are partitioned into states to keep, x_1, and states to eliminate, x_0

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_0 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{10} \\ A_{01} & A_{00} \end{bmatrix} \begin{bmatrix} x_1 \\ x_0 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_0 \end{bmatrix} u$$
$$y = \begin{bmatrix} C_1 & C_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_0 \end{bmatrix} + Du \tag{1.5}$$

If the states x_0 are simply truncated, one gets a reduced-order model that corresponds well with the original model at higher frequencies.

$$\dot{x}_1 = A_{11} x_1 + B_1 u$$
$$y = C_1 x_1 + Du$$

To instead get a model that corresponds well to the original model at lower frequencies, Singular Perturbation Approximation can be used. Setting $\dot{x}_0 = 0$ in the first row of (1.5), solving that for x_1 and substituting into the other rows gives the reduced-order model. See Johansson (1993) for details.

Trajectory specific model reduction of chemical systems Trajectory-Specific model reduction of linear systems is done in Mavrikis and Vinter (1997). A generalization of Balanced truncation is used to find a reduced order model for a class of input signals. Stability is proved. Extensions for nonlinear systems are done in Mavrikis (1997).

Model reduction of Nonlinear systems using Perturbation Theory

There are several approaches to nonlinear model reduction. One that is closely related to the ideas in this thesis is the so called *perturbation method*, described in Khalil (1996). The purpose of the method is to find an approximation of a model where the parameter dependence is simpler. The complexity of the parameter dependence but not the model order is reduced.

Khalil considers systems on the form

$$\dot{x} = f(t, x, \varepsilon) \tag{1.6}$$

with a solution $x(t, \varepsilon)$ with a complicated dependence on ε. The trajectory is written as a Taylor expansion in ε with $x_k(t)$ as coefficients. The coefficients $x_k(t)$ are all independent of ε and can be computed separately.

$$x(t, \varepsilon) = \sum_{k=0}^{\infty} x_k(t)\varepsilon^k \qquad \dot{x}_k = f_k(t, x_k) \tag{1.7}$$

The question is now how to find the functions f_k. This is done by identifying coefficients in the Taylor expansion of the derivative of (1.6) around $\varepsilon = 0$.

$$\sum_{k=0}^{\infty} \dot{x}_k(t)\varepsilon^k = \dot{x}(\varepsilon, t) = f(t, x(t, \varepsilon), \varepsilon) = \sum_{k=0}^{\infty} \left.\frac{\partial^k f(t, x(t, \varepsilon), \varepsilon)}{\partial \varepsilon^k}\right|_{\varepsilon=0} \varepsilon^k$$

Substituting $x(t)$ with the sum in (1.7) and identifying coefficients for each power of ε gives

$$\dot{x}_k(t) = \left.\frac{d^k}{d\varepsilon^k} f(t, x(t, \varepsilon), \varepsilon)\right|_{\varepsilon=0}$$

Solving this for $k = 0$ gives $x_0(t)$. After substituting $x_0(t)$ one can then solve for all $x_k(t)$. If the initial condition $x(t_0)$ does not depend on ε, then the initial conditions for $x_k(t)$ can easily be chosen as $x_0(t_0) = x(t_0), x_1(t_0) = 0, \ldots, x_N(t_0) = 0$. If $x(t_0)$ is depending on ε one can do a Taylor expansion around $\varepsilon = 0$ similar to (1.7) to identify the different $x_k(t_0, \varepsilon)$.

For large N the remainder term is small and the solution $x(t, \varepsilon)$ to (1.6) is then approximated as

$$x(t, \varepsilon) = \sum_{k=0}^{\infty} x_k(t)\varepsilon^k \approx \sum_{k=0}^{N-1} x_k(t)\varepsilon^k \tag{1.8}$$

It is shown that the error is bounded by

$$x(t, \varepsilon) - \sum_{k=0}^{N-1} x_k(t)\varepsilon^k = O(\varepsilon^N)$$

but no quantitative bound on the error is given. It would be interesting to use the error bound ideas presented in Chapter 4 to investigate the error caused by the perturbation method for different values on N. Then one could, before the reduction, obtain a number N that will result in a sufficiently small error.

With regard to the perturbation discussions Khalil also treats systems on the form

$$\dot{x} = f(t, x, z, \varepsilon)$$
$$\varepsilon\dot{z} = g(t, x, z, \varepsilon)$$

and

$$\dot{x} = \varepsilon f(t, x, \varepsilon)$$

Such systems are not considered in this thesis.

Trajectory-Specific Model Reduction of Chemical Systems

In Petzold and Zhu (1997) nonlinear models for chemical reactions are studied. The purpose of the paper is to find a reduced model that captures the important reactions but ignores the unimportant ones. The reduced model is of the same order as the original one, but less complex.

Consider a system with n chemical species y_i and N reactions $F_i(y)$. Typically $N >> n$. The models are written in the explicit form as

$$\dot{y} = SF(y) \tag{1.9}$$

where $y = [\, y_1 \dots y_n \,]^T$ and $F(y) = [\, F_1(y) \dots F_N(y) \,]^T$. The amount of each species consumed or produced by each reaction is determined by the coefficients $S = [\, S_1 \dots S_N \,]$, where each S_i is a column vector. A specific trajectory is chosen by choosing the initial conditions $y(0) = y_0$. A specified time interval is studied $0 \leq t \leq b$. A reduced model looks like

$$\dot{z} = SDF(z) \tag{1.10}$$

where $z = [\, z_1 \dots z_n \,]^T$ and $D = \mathrm{diag}(d_1 \dots d_N)$ with each d_i equal to 0 or 1 representing that a reaction $F_r(z)$ should be neglected or not. The user has to decide how many reactions $0 \leq k \leq N$ that should be neglected, but then the algorithm decides which reactions. The model reduction problem is formulated as an optimization problem.

$$\begin{aligned} &\min \|y - z\| \\ &\dot{y} = SF(y) && y(0) = y_0 \\ &\dot{z} = SDF(z) && z(0) = y_0 \\ &\sum_{i=1}^{N} d_i = k \end{aligned}$$

Since solving the discrete optimization problem is hard, it is reformulated as a continuous optimization problem and then the d_i's found are rounded to 0 or 1. To avoid large rounding error, a penalty function is

added to the optimization problem that forces the coefficients d_i to be close to 0 or 1, e.g.,

$$g(d_1, \ldots, d_N) = \sum_{i=1}^{N} (d_i - {d_i}^2)^\beta \leq r$$

In the paper it is further discussed how to choose the initial conditions of d_i. Starting with all $d_i = 1$ is the easiest choice but there are more advanced ones. The optimization problem is solved with a package called DASOPT using nonlinear programming techniques.

It would be interesting to combine this methods with the ideas presented in this thesis. The method described by Petzold could be used to determine which parameters to neglect and then the ideas from this thesis, in an extended version, could be used to adjust the remaining parameters and to give bounds on the error.

The paper also discusses reduction of the number of species, i.e., model order reduction, but that is not as closely related to the work presented in this thesis.

Nonlinear system identification

Model reduction and system identification are closely related. Nonlinear system identification is divided in Sjöberg and et al (1995) according to how well the model structure of the model that is to be identified is known. The categories are

- White-box
- Grey-box
- Black-box

Identification methods can also be classified into identification of

- Linear models
- Nonlinear models

An overview of nonlinear stochastic grey-box identification, also called probabilistic semi-physical model design, is given in

Bohlin and Graebe (1994). The paper is dealing with identification of the parameters θ in models on the form

$$\begin{aligned} x(t) &= x_0 + \int_{t_0}^{t} m(\tau, x, u, p, \theta)d\tau + \int_{t_0}^{t} \sigma(t, p, \theta)d\beta \\ y(t_k) &= h(t_k, x, u, p, \theta) + R(t_k, \theta)w(t_k) \end{aligned} \tag{1.11}$$

The parts σ and $w(t)$ represent stochastics and the basis for the identification is measured data points of the y and u.

The problem is similar to one studied in Chapter 2, where, however, the data points come from simulations and there is no noise and stochastics involved. The goal there is to identify the parameters ε in the equation (2.7) describing the norm of the state error.

$$f(\varepsilon, T) = \int_0^T \sum_i \tilde{x}_i(t)^2 dt \tag{1.12}$$

Parameters in (1.11) are identified using the maximum likelihood method. A likelihood function $L(\theta)$ is defined depending on the parameters to identify θ and statistical properties of σ and w. The Jacobian $G(\theta) = \partial L/\partial\theta$ and Hessian $H(\theta) = \partial^2 L/\partial\theta^2$ are estimated and then the parameters θ are found via Newton iterations

$$\hat{\theta}_{j+1} = \hat{\theta}_j - \hat{H}(\hat{\theta}_j)^{-1}\hat{G}(\hat{\theta}_j)$$

This maximization of a likelihood function looks similar to the minimization of the norm of the error (1.12), done in Chapter 2, equation (2.16), also using the Newton direction.

$$\varepsilon_{opt} \approx -H(0, T)^{-1}G(0, T) \tag{1.13}$$

1.4 Outline of thesis

The Part I of the thesis is organized as follows. The reduction principles are developed in Chapter 2 and exemplified via a rotating pendulum.

A drum boiler model is presented and simplified in Chapter 3. Calculation of strict upper bounds of the error are done in Chapter 4. The conclusions are summarized in Chapter 5.

The Part II of the thesis contains the paper "Implementation aspects of the PLC standard IEC 1131-3".

2

Model Reduction

2.1 Introduction

In this chapter some ideas for model reduction of nonlinear systems are presented. The ideas are exemplified using a rotating inverted pendulum.

This chapter is organized as follows. The problem is formulated in Section 2.2 and exemplified by the pendulum in Section 2.3. The approximation of the state error is described in Section 2.4 and Section 2.5. Optimization of the remaining terms are done in Section 2.6. This is then applied to the pendulum example in Section 2.7.

2.2 Problem formulation

This chapter deals with nonlinear models that can be written on the form

$$0 = F(\dot{x}(t), x(t)) \tag{2.1}$$

Suppose that the model contains a number of parameters $a_0, \ldots, a_m$.

$$0 = F(\dot{x}(t), x(t), a_0, \ldots, a_m) \tag{2.2}$$

Assuming that each parameter a_i has nominal value one, the nominal model is

$$0 = F(\dot{x}(t), x(t), 1, \dots, 1) \tag{2.3}$$

The formulation (2.2) is chosen so that the reduced model is written as

$$0 = F(\dot{\bar{x}}(t), \bar{x}(t), 0, 1, \dots, 1) \tag{2.4}$$

The nonzero parameters can be adjusted with some coefficients $\varepsilon_1, \dots, \varepsilon_m$ to compensate for the omitted parts. The reduced and compensated model is written as

$$0 = F(\dot{\bar{x}}(t), \bar{x}(t), 0, 1 + \varepsilon_1, \dots, 1 + \varepsilon_m) \tag{2.5}$$

This is closely related to system identification. One has to be aware of that, depending on how the parameters a_i are chosen, some physical interpretation of the model might get lost.

The same notation $\bar{x}(t)$ is used for trajectory of the reduced model (2.4) and the trajectory of the model that is reduced and compensated (2.5). The error of the states is in both cases denoted

$$\tilde{x}(t) = x(t) - \bar{x}(t) \tag{2.6}$$

The goal is to estimate the error $\tilde{x}(t)$ and to minimize the norm of the error $\|\tilde{x}(t)\|$ over the time interval $[0, T]$ with respect to ε. With e defined as

$$e(\varepsilon, T) = \|\tilde{x}(t)\| = \int_0^T \sum_i \tilde{x}_i(t)^2 dt \tag{2.7}$$

one has to minimize

$$\min_{\varepsilon} e(\varepsilon, T)$$

Expressions for $\tilde{x}(t)$ as a function of *eps* and $e(\varepsilon)$ is needed. The trajectory $\bar{x}(t)$ is not known, since ε is not known. Therefore a differential equation defining $\tilde{x}$ is derived and $\tilde{x}$ is then simulated approximately. This is done in Section 2.4. The optimization is then done in Section 2.6.

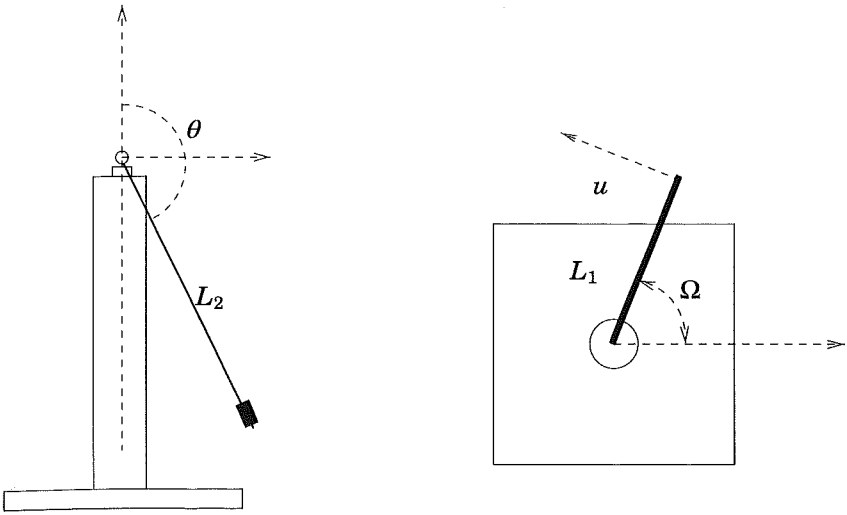

Figure 2.1 The rotating pendulum.

2.3 Pendulum example

In this section the models for a rotating and a non-rotating pendulum is compared.

Rotating pendulum

A rotating pendulum is shown in Figure 2.1. A pendulum of the length L_2 is hanging at the end of rotating bar with the length L_1. The acceleration of end point of the rotating bar is considered control signal u.

The pendulum is described by

$$\dot{x} = f(x, u)$$

where

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \\ \Omega \\ \dot{\Omega} \end{bmatrix}$$

and

$$f(x,u) = \begin{bmatrix} x_2 \\ \frac{g}{L_2}\sin x_1 + \frac{x_4^2}{2}\sin 2x_1 - \frac{u}{L_2}\cos x_1 \\ x_4 \\ \frac{u}{L_1} \end{bmatrix} \tag{2.8}$$

Non-rotating pendulum

A non-rotating pendulum is shown in Figure 2.2. The pendulum is hanging on a cart and the acceleration of the cart is considered as control signal u The pendulum is described by partly the same equations as the rotating one:

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

$$f(x,u) = \begin{bmatrix} x_2 \\ \frac{g}{L_2}\sin x_1 - \frac{u}{L_2}\cos x_1 \end{bmatrix}$$

Simplification goal

The purpose is to investigate how well the rotating pendulum can be described by a non-rotating one. The state Ω is not considered at all. Both models can now be written as (2.2) ... (2.5) with

$$0 = f(x,u) - \dot{x} = F(\dot{x}(t), x(t), a_0, \ldots, a_4) \tag{2.9}$$

where

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \\ \dot{\Omega} \end{bmatrix}$$

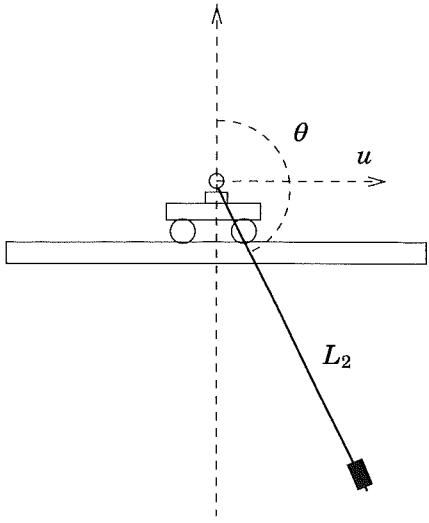

Figure 2.2 The non-rotating pendulum.

$$f(x,u) = \begin{bmatrix} a_3 x_2 \\ a_0 \frac{x_3^2}{2} \sin 2x_1 + a_1 \frac{g}{L_2} \sin x_1 - a_2 \frac{u}{L_2} \cos x_1 \\ a_4 \frac{u}{L_1} \end{bmatrix} \tag{2.10}$$

Setting all $a_i = 1$ gives the rotating pendulum and setting $a_0 = 0$ gives the non-rotating one. The model can then be adjusted with the parameters $a_1 = 1 + \varepsilon_1, \ldots, a_4 = 1 + \varepsilon_4$. This is done in when the pendulum example is continued in Section 2.7.

2.4 Error approximation

In this section approximate expressions are derived from which $\tilde{x}(t)$ in (2.6) can be simulated. The expressions are derived for the reduced and compensated model (2.5), but setting $\varepsilon = 0$ gives expressions for the reduced model (2.4). A simulation of a trajectory $x(t)$ from (2.1) or a trajectory $\bar{x}(t)$ from (2.4) or (2.5) is needed to evaluate the linearization.

The model (2.2) is linearized around the interesting trajectory. The more linear the model F is and the smaller the error $\tilde{x}(t)$ is, the more accurate the approximation is. Subtracting (2.5) from (2.3) and differentiating gives

$$\begin{aligned} 0 = F(\dot{x}, x, 1, \dots, 1) - F(\dot{\bar{x}}, \bar{x}, 0, 1+\varepsilon_1, \dots, 1+\varepsilon_m) \\ \approx \frac{\partial F}{\partial \dot{x}}\dot{\tilde{x}} + \frac{\partial F}{\partial x}\tilde{x} + \frac{\partial F}{\partial a_0} + \sum_{k=1}^{m} \frac{\partial F}{\partial a_k}(-\varepsilon_k) \end{aligned} \quad (2.11)$$

The partial derivatives of F can be evaluated either at $(\dot{x}, x, 1, \dots, 1)$ or at $(\dot{\bar{x}}, \bar{x}, 0, 1+\varepsilon_1, \dots, 1+\varepsilon_m)$. The first trajectory can be used if the numerical values of ε are not known at this stage. If, however, they are known, or they are zero, the second trajectory can be used. This is interesting simulation of the original model is time-consuming. If $\frac{\partial F}{\partial \dot{x}}$ is invertible (2.11) can then be written as

$$\dot{\tilde{x}}(t) \approx \left(\frac{\partial F}{\partial \dot{x}}\right)^{-1} \left(-\frac{\partial F}{\partial x}\tilde{x}(t) - \frac{\partial F}{\partial a_0} + \sum_{k=1}^{m} \frac{\partial F}{\partial a_k}\varepsilon_k\right) \quad (2.12)$$

With this equation the state error $\tilde{x}(t)$ defined in (2.6) can be simulated if the original model is known, the reduced model is known and a trajectory from either of them is available.

2.5 Error contribution of each term

This section studies Equation (2.12) in detail. It is investigated how the different parts of a model (2.2), associated with the different a_i, contributes to the error $\tilde{x}(t)$ in (2.6) when (2.3) is approximated by (2.5). A notation is introduced that in Section 2.6 will give a closed-form analytic solution the problem of optimizing ε. The purpose is to find a notation $d_k(t)$ with which the $\tilde{x}(t)$ can be written as

$$\tilde{x}(t) \approx d_0(t) - \sum_{k=1}^{n} \varepsilon_k d_k(t) \quad (2.13)$$

Equation (2.12) is a differential equation with $m+1$ input signals. Since the errors initial condition is zero, the superposition principle gives that the system can be simulated for each input signal separately and then the solutions can be added to obtain the solution of (2.12). The solutions to the system simulated with each input is given the notation $d_k(t)$. The different $d_k(t)$ are defined by

$$\dot{d}_k(t) = \left(\frac{\partial F}{\partial \dot{x}}\right)^{-1}\left(-\frac{\partial F}{\partial x}d_k(t) - \frac{\partial F}{\partial a_k}\right)$$

with $d_k(0) = 0$. Each $d_k(t)$ can now be simulated separately. With this notation $\tilde{x}(t)$ is given by (2.13).

2.6 Optimization

The purpose of this section is to find an optimal value of the coefficients ε in (2.5) to minimize the norm of the state error $\tilde{x}$ defined by (2.6) and (2.7). The original model (2.3), its simulated trajectory $x(t)$, and the structure of the compensated model (2.5) must be available. To find a closed-form expression of the function $e(\varepsilon)$ to minimize, the notation c_{kl} is introduced. Each c_{kl} is defined as follows

$$\begin{aligned} e(\varepsilon, T) = \|\tilde{x}(t)\| \approx \left\| d_0(t) - \sum_{k=1}^{m} \varepsilon_k d_k(t) \right\| = \\ = \int_0^T \sum_i \left(d_{0,i}(t) - \sum_{k=1}^{m} \varepsilon_k d_{k,i}(t) \right)^2 dt = \\ = \underbrace{\int_0^T d_0(t)^T d_0(t) dt}_{c_{00}} - 2\sum_{k=1}^{m} \varepsilon_k \underbrace{\int_0^T d_0(t)^T d_k(t)) dt}_{c_{k0}} + \\ + \sum_{k=1}^{m}\sum_{l=1}^{m} \varepsilon_k \varepsilon_l \underbrace{\int_0^T d_k(t)^T d_l(t) dt}_{c_{kl}} \end{aligned}$$

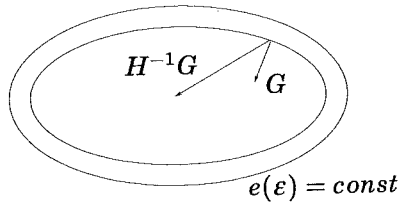

Figure 2.3 The newton direction $H^{-1}G$ of a nonlinear optimization problem.

With this notation $e(\varepsilon, T)$ in (2.7) can then be written as

$$e(\varepsilon, T) \approx c_{00} - 2\sum_{k=1}^{m} \varepsilon_k c_{0k} + \sum_{k=1}^{m}\sum_{l=1}^{m} \varepsilon_k \varepsilon_l c_{kl}$$

and with the matrix notation

$$\left[\begin{array}{c|c} c_{00} & c_0^T \\ \hline c_0 & C \end{array}\right] = \left[\begin{array}{c|ccc} c_{00} & c_{01} & \cdots & c_{0m} \\ \hline c_{10} & c_{11} & \cdots & c_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1m} & c_{m1} & \cdots & c_{mm} \end{array}\right] \tag{2.14}$$

as

$$e(\varepsilon, T) \approx c_{00} - 2c_0^T \varepsilon + \varepsilon^T C \varepsilon \tag{2.15}$$

The nonlinear optimization problem is now formulated as a linear optimization problem using linearization around a trajectory. This problem will be solved by going one step in the newton direction, illustrated in Figure 2.3. This gives the following expression for an optimal ε, involving the Jacobian and Hessian of e.

$$\min_{\varepsilon} e(\varepsilon, T) \quad \Rightarrow \quad \varepsilon_{opt} = -H(0,T)^{-1} G(0,T) \tag{2.16}$$

The Jacobian of e is calculated:

$$G(\varepsilon,T)=\frac{\partial e(\varepsilon,T)}{\partial \varepsilon}=\begin{bmatrix}-2c_{10}+2\sum_{i=0}^{n}\varepsilon_i c_{1i}\\ \vdots\\ -2c_{n0}+2\sum_{i=0}^{n}\varepsilon_i c_{ni}\end{bmatrix}$$
$$=-2c_0+2C\varepsilon$$

Setting $\varepsilon=0$ gives

$$G(0,T)=-2c_0$$

The Hessian is given by

$$H(\varepsilon,T)=\frac{\partial^2 e(\varepsilon,T)}{\partial \varepsilon^2}=\begin{bmatrix}2c_{11} & \dots & 2c_{1n}\\ \vdots & \ddots & \vdots\\ 2c_{n1} & \dots & 2c_{nn}\end{bmatrix}=2C$$

The conclusion is that, assuming that (2.15) is correct, the optimal way to choose the parameters ε in (2.5) is

$$\varepsilon_{opt}=-H(0,T)^{-1}G(0,T)=C^{-1}c_0 \tag{2.17}$$

Line search

Due to approximations in (2.11) it is not known how close to optimum this choice of ε is. A possible way to find a an ε closer to optimum is to do a small line search along the newton direction (see Figure 2.3). The system (2.5) is simulated with $\varepsilon_{new}=k\varepsilon$ close to the original ε given by (2.17). This is done for a number of different values of k. In the examples in this thesis, 11 different k:s between 0.7 and 1.3 are used. For each simulation the error $e(k\varepsilon)$ given by (2.7) is found by comparing the simulations of (2.3) and (2.5). The value of k giving the minimum value of $e(k\varepsilon)$ is chosen and ε is adjusted:

$$\varepsilon_{new}=k\varepsilon \tag{2.18}$$

2.7 Pendulum example continued

In this section the algorithm developed in Section 2.4 ... Section 2.6 is applied to the pendulum example. it is shown that the pendulum

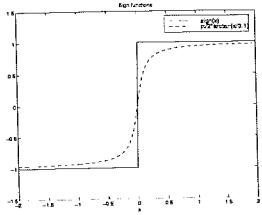

Figure 2.4 The sgn function is approximated by the arctan function.

can be approximated with a non-rotating one, with somewhat different parameters.

Choice of control law

The purpose is to swing up the rotating pendulum (2.9) with all $a_i = 1$ from downwards ($\theta = \pi$) to upwards ($\theta = 0$) position. This can be done by applying a control signal

$$u(x) = 2g\left(\sin\sqrt{\frac{L_2}{g}}x_2 + \operatorname{sgn}(\cos x_1)\sin x_1\right) + x_3^2 L_2 \sin x_1 + L_2 x_3$$

To avoid the problems of taking the derivative of the sgn function it is approximated by the arctan function, illustrated by Figure 2.4. The control law used is

$$u(x) = 2g\left(\sin\sqrt{\frac{L_2}{g}}x_2 + \frac{2}{\pi}\arctan\frac{\cos x_1}{0.1}\sin x_1\right) + x_3^2 L_2 \sin x_1 + L_2 x_3 \quad (2.19)$$

The control law (2.19) is shown in Figure 2.5.

Simulations

A simulation of the rotating pendulum, (2.9) with all $a_i = 1$, is shown by the line (—) in Figure 2.6. The control signal (2.19) is shown in Figure 2.5. The choice of parameters was.

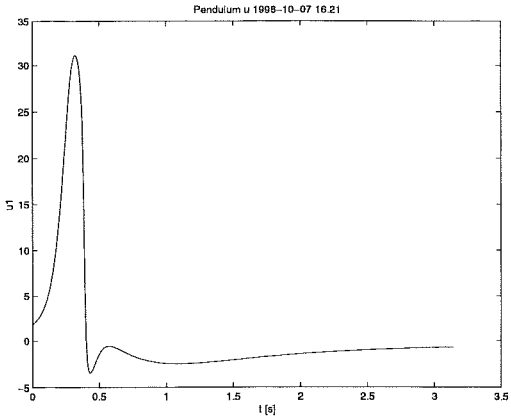

Figure 2.5 Control signal (2.19) used to swing up the pendulum.

L_1	$1\ m$
L_2	$0.3\ m$
g	$9.81\ m/s^2$

This results in a trajectory shown by the solid line in Figure 2.6. The non-rotating pendulum is now simulated, obtained by setting $a_0 = 0$ in (2.9). The resulting trajectory (- -) is also shown in Figure 2.6.

To be able to simulate the error $\tilde{x}$ using (2.12) some expressions, e.g. the Jacobian is needed.

Jacobian

The derivative of (2.9) is taken with respect to x, $\dot{x}$, and a_i.

$$\left.\frac{\partial F}{\partial x}\right|_{a_i=1} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{\partial F_2}{\partial x_1} & 0 & x_3^2 \sin 2x_1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -\cos x_1 \\ \frac{1}{L_2} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x_1} & \frac{\partial u}{\partial x_2} & \frac{\partial u}{\partial x_3} \end{bmatrix} \tag{2.20}$$

where

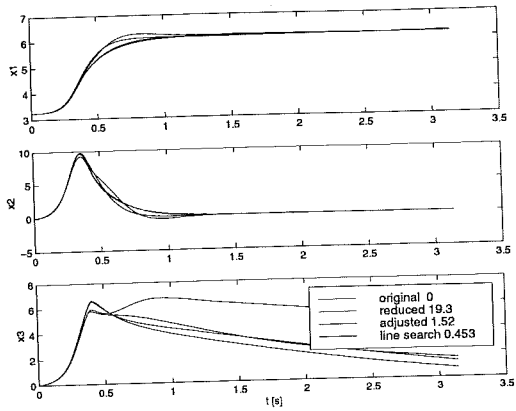

Figure 2.6 Simulation of the pendulum (2.10), the reduced model and the compensated model, and the model compensated with the result of the line search. The state error e in (2.7) for the different cases is written in the box.

$$\frac{\partial F_2}{\partial x_1} = \left(\frac{g}{L_2}\cos x_1 + x_3^2 \cos 2x_1 + \frac{u}{L_2}\sin x_1\right)$$

$$\begin{aligned}\frac{du}{dx_1} &= 2g\frac{2}{\pi}\arctan\frac{\cos x_1}{0.1}\cos x_1 \\ &\quad - 2g\frac{2}{\pi}\frac{\cos x_1}{x^2+\cos^2 x_1}\sin^2 x_1 \\ &\quad + x_3^2 L_2 * \cos x_1\end{aligned}$$

$$\frac{du}{dx_2} = 2g\cos\left(\frac{L_2}{g}x_2\right)\sqrt{\frac{L_2}{g}}$$

$$\frac{du}{dx_3} = 2x_3 L_2 \sin x1 + L_2$$

Further, (2.9) gives

$$\frac{\partial F}{\partial \dot{x}} = -I$$

and (2.9) and (2.10) gives

$$\frac{\partial F}{\partial a} = \begin{bmatrix} 0 & 0 & 0 & x_2 & 0 \\ \frac{x_3^2}{2}\sin 2x_1 & \frac{g}{L_2}\sin x_1 & -\frac{u}{L_2}\cos x_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{u}{L_1} \end{bmatrix}$$

with $a = [a_0 \dots a_4]^T$.

Simplification

The error $\tilde{x}$ is now simulated using (2.12), with all $\varepsilon_i = 0$. The simulated error is shown in Figure 2.7 and compared with the actual error, i.e., the difference between the trajectories (—) and (- -) in Figure 2.6. The difference between the curves is explained by the approximations associated with (2.12). The similarity between the curves verifies the derivations of the formulas in Section 2.4 and shows that the approximation errors are not to large.

Compensation

The goal is now to adjust the parameters $a_i = 1+\varepsilon_i$ in (2.9) to minimize the error. Applying (2.14) gives

$$\left[\begin{array}{c|c} c_{00} & c_0^T \\ \hline c_0 & C \end{array}\right] = \left[\begin{array}{r|rrrr} 15.81 & 50.04 & -54.88 & 5.21 & 3.27 \\ \hline 50.0 & 174 & -208 & 17.2 & 6.03 \\ -54.9 & -208 & 289 & -4.41 & -6.51 \\ 5.21 & 17.2 & -4.41 & 25.2 & -2.86 \\ 3.27 & 6.03 & -6.51 & -2.86 & 9.89 \end{array}\right]$$

and (2.17) gives

$$\varepsilon = \begin{bmatrix} 0.4749 \\ 0.1532 \\ -0.0781 \\ 0.1187 \end{bmatrix} \tag{2.21}$$

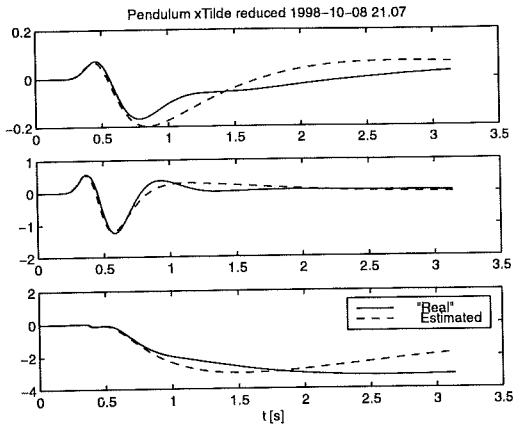

Figure 2.7 The state error $\tilde{x}$ for the reduced pendulum, simulated using (2.12), with all $\varepsilon_i = 0$ (- -) , compared with the difference between the simulations of the nominal and reduced model (—).

The resulting trajectory when (2.21) is applied like in (2.5) is the line (...) in Figure 2.6.

Line search

A line search is done as described in Section 2.6. The values of k and the corresponding errors are shown in Figure 2.8. The best choice among the value of k seem to be 0.9. Therefore the ε is changed:

$$\varepsilon_{new} = 0.9\varepsilon = \begin{bmatrix} 0.4274 \\ 0.1378 \\ -0.0703 \\ 0.1069 \end{bmatrix} \tag{2.22}$$

A simulation with ε_{new} is shown in Figure 2.6 (.-.-).

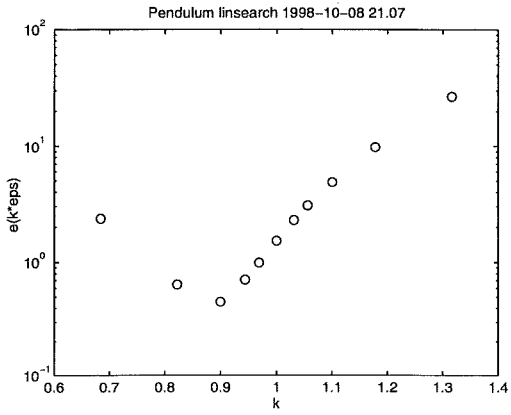

Figure 2.8 A line search along the newton direction to find a good value of ε for the pendulum example, is done as described in Section 2.6. The error ε is plotted for different values of k in (2.18).

Reduced model

A reduced and compensated pendulum model is given by

$$f(x,u) = \begin{bmatrix} a_3 x_2 \\ a_1 \frac{g}{L_2} \sin x_1 - a_2 \frac{u}{L_2} \cos x_1 \\ a_4 \frac{u}{L_1} \end{bmatrix}$$

with

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \\ \dot{\Omega} \end{bmatrix}$$

and $a_i = 1 + \varepsilon_i$ with ε given by (2.22).

Physical interpretation

The reduced pendulum model corresponds to a non-rotating pendulum. The adjustments can also be interpreted physically. The ε_3 is a scaling

in time that will affect also the other states. The ε_1 can be interpreted as smaller value of L_2, i.e., shorter pendulum, while ε_2 corresponds to a smaller control signal u. The ε_4 value affects only the third state, since the states are decoupled in the reduced model. The conclusion is that the reduced and compensated model is a non-rotating pendulum with other numerical values on the physical parameters.

Concluding remarks

This chapter has presented ideas about how to estimate the error of the states caused by model reduction in Section 2.4. Further it is discussed in Section 2.5 and Section 2.6 how remaining parts of a model can be adjusted to minimize the error. Both ideas are applied to a pendulum example in Section 2.7. Simulation of the original and reduced models are shown in Figure 2.6. The error is rather small, indicating that a rotating pendulum can be quite well approximated by a pendulum on a cart, for a swing up trajectory. The error is reduced significantly by adjusting remaining parameters, shown by (...) in Figure 2.6. A line search shown in Figure 2.8 reduces the error further, see (_._.) in Figure 2.6.

3

Drum Boiler Example

3.1 Introduction

In this chapter, second and fourth order nonlinear models of a drum boiler model are studied. The boiler models were presented in Bell and Åström (1998) and Bell and Åström (1996). Figure 3.1 shows a schematic picture of the boiler.

Input signals used in identification experiments of a boiler at Öresundsverket, Malmö are used. It is shown that for this trajectory it is possible to reduce large parts of the model, adjust the remaining parameters and get a model that gives only a rather small error in the states.

The second order model is presented in Section 3.2. The reduction ideas from Chapter 2 is applied in Section 3.3 and the resulting reduced model is presented in Section 3.4. A similar reduction procedure is applied to the fourth order model in Section 3.5.

3.2 Second order model

In the second order model, the feed-water flow q_f, the steam flow q_s and the amount of fuel Q are considered as input signals. The states are the volume of water V_{wt} and the pressure p in the drum. The states are considered as output signals. The following equations describe the

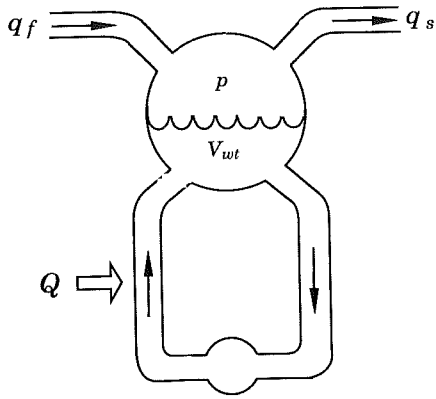

Figure 3.1 A schematic picture of the drum boiler. The feed-water flow q_f, the steam flow q_s and the amount of fuel Q are considered as input signals. In the second order model, the volume of water V_{wt} and the pressure p in the drum are considered as output signals.

dynamics of the boiler.

$$\begin{aligned} e_{11}\frac{dV_{wt}}{dt} + e_{12}\frac{dp}{dt} &= q_f - q_s \\ e_{21}\frac{dV_{wt}}{dt} + e_{22}\frac{dp}{dt} &= Q + q_f h_f - q_s h_s \end{aligned} \tag{3.1}$$

The first equation is a mass balance, while the second is an energy balance. The terms

$$\begin{aligned} e_{11} &= \rho_w - \rho_s \\ e_{12} &= V_{st}\frac{\partial \rho_s}{\partial p} + V_{wt}\frac{\partial \rho_w}{\partial p} \end{aligned}$$

represents mass storage while

$$\begin{aligned} e_{21} &= \rho_w h_w - \rho_s h_s \\ e_{22} &= V_{st}\left(h_s\frac{\partial \rho_s}{\partial p} + \rho_s\frac{\partial h_s}{\partial p}\right) \\ &\quad + V_{wt}\left(h_w\frac{\partial \rho_w}{\partial p} + \rho_w\frac{\partial h_w}{\partial p}\right) + m_t C_p\frac{\partial t_s}{\partial p} \end{aligned}$$

represents energy storage. The model has a bilinear structure and hence can be written as

$$0 = F(\dot{x}, x) = B(x)u - E(x)\dot{x}$$

where

$$B(x) = \begin{bmatrix} 0 & 1 & -1 \\ 1 & h_{fw} & -h_s \end{bmatrix} \qquad u = \begin{bmatrix} Q \\ q_f \\ q_s \end{bmatrix}$$

$$h_{fw} = 10^3 \left(C_{fw} T_{fw} + p \frac{10^3}{\rho_w} \right)$$

$$E(x) = \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix} \qquad x = \begin{bmatrix} V_{wt} \\ p \end{bmatrix}$$

Simulation

The inputs u shown in Figure 3.2 are experimental data from Öresundsverket. The pressure in Figure 3.3 are measured at the same experiment and compared with a simulation of the model (3.1). Steam tables are required to evaluate variables depending on the pressure, e.g., h_s, h_w, ρ_s and ρ_w. Quadratic functions have been used to represent the steam tables. The variables depending on the pressure, e.g., ρ_s and h_s are approximated by second order polynomials of the pressure p. The parameters used were

V_t	40 m^3
m_t	325 000 kg
C_p	550
C_{fw}	4.18

3.3 Reduction

The boiler model is reduced in several steps. After studying which parameters that are likely to be possible to neglect the reduced model is simulated. Then an optimization adjusts the remaining parameters. Since the optimization contain approximations, a line search along the Newton direction is performed.

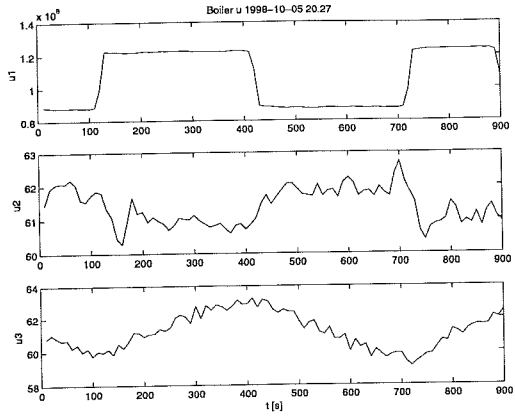

Figure 3.2 Input signals $u^T = [Q \; q_f \; q_s]$ from the measurements at Öresundsverket.

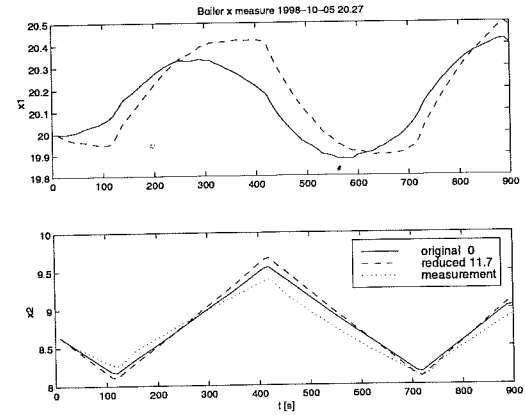

Figure 3.3 Simulation of the boiler model original (3.2) with all $a_{ij} = 1$ and the reduced model with $a_0 = 0$, all with the input signals shown in Figure 3.2. The state error e in (2.7) is written in the box. The second state $x_2 = p$ is compared with measurements.

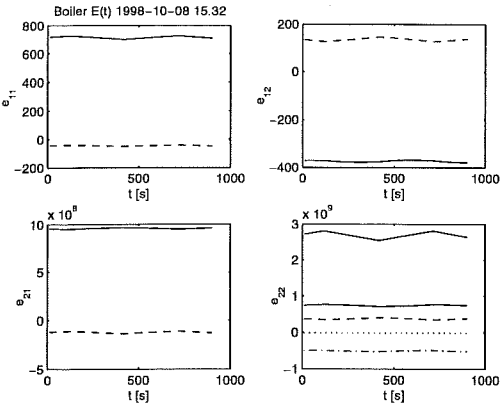

Figure 3.4 Additive components of each element e_{ij} in E. The largest component (—) in each $e_{ij}(t)$ is kept and the others are neglected by putting an a_0 in front of them in (3.2).

Simplifications goal

In this section it is studied whether it is possible to delete any part the components e_{ij}. Figure 3.4 shows each e_{ij} term composed by sub-terms. This plot provides the basis for choosing which component of each e_{ij} to neglect, i.e., multiply by a_0. Based on the relative magnitudes shown in Figure 3.4 the following hypothesis for model reduction can be made. Try to keep the largest sub-term, plotted with a solid line in Figure 3.4 and neglect the others. The e_{ij} terms can then be written as

$$
\begin{aligned}
e_{11} &= a_1\rho_w - a_0\rho_s \\
e_{12} &= a_0 V_{st}\frac{\partial \rho_s}{\partial p} + a_2 V_{wt}\frac{\partial \rho_w}{\partial p} \\
e_{21} &= a_3\rho_w h_w - a_0\rho_s h_s \\
e_{22} &= a_0 V_{st}(h_s\frac{\partial \rho_s}{\partial p} + \rho_s\frac{\partial h_s}{\partial p}) + a_0 V_{wt}(h_w\frac{\partial \rho_w}{\partial p} + \rho_w\frac{\partial h_w}{\partial p}) \\
&\quad + a_4 m_t C_p\frac{\partial t_s}{\partial p}
\end{aligned}
\tag{3.2}
$$

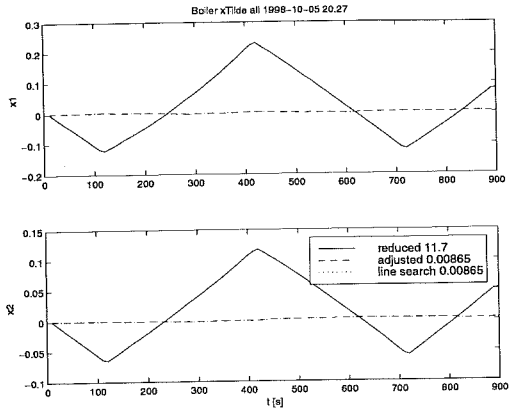

Figure 3.5 The state error $\tilde{x}$ from simulations of the reduced boiler model, compensated model, and the model compensated with the result of the line search. The state error e in (2.7) for the different cases is written in the box.

To use the notation of (2.1), the boiler model can be written as

$$0 = F(\dot{x}, x, a_1, \ldots, a_m) = E(x, a)^{-1}\dot{x}(t) - B(x)u \tag{3.3}$$

The Jacobian involve quite complex expressions, but is straight forward to calculate.

Simplification

The error $\tilde{x}$ simulated from (2.12), with all $\varepsilon_i = 0$, is shown in Figure 3.5 and compared with the actual error, i.e., the difference between the trajectories (—) and (- -) in Figure 3.3.

A simulation of the reduced model is shown by the line (- -) in Figure 3.3.

Adjustment

The next step is now to compensate the remaining terms in the model. Following the ideas and notation presented in Section 2.5 and Sec-

tion 2.6 gives

$$\left[\begin{array}{c|c} c_{00} & c_0^T \\ \hline c_0 & C \end{array}\right] = \left[\begin{array}{r|rrrr} 12.0 & -45.9 & 15.56 & -4.28 & -37.09 \\ \hline -9.45 & 51.7 & -31.1 & 3.60 & 11.2 \\ 15.6 & -31.1 & 43.0 & -1.96 & -13.3 \\ -4.28 & 3.60 & -1.96 & 6.67 & 22.2 \\ -37.1 & 11.2 & -13.3 & 22.2 & 173 \end{array}\right]$$

Applying (2.17) gives

$$\varepsilon_{opt} = C^{-1}c_0 = \begin{bmatrix} 0.0648 \\ 0.3504 \\ 0.1100 \\ -0.2055 \end{bmatrix} \tag{3.4}$$

The simulation of the compensated model is shown in Figure 3.3 (...). The error $\tilde{x}$ simulated from (2.12), with all $\varepsilon_i = 0$, is shown in Figure 3.5.

Line search

Also for the boiler example a line search along the newton direction is done as described in Section 2.6. The values of k and the corresponding errors $e(k\varepsilon)$ are shown in the Figure 2.8. The best choice among the value of k seem to be 1.00. Therefore ε is not changed and ε still equals to (3.4) and therefore the curves (_ _) and (_._.) are on top of each other.

3.4 Results

The model obtained after reduction, adjustment and line search looks like

$$a_1\rho_w\frac{dV_{wt}}{dt} + a_2V_{wt}\frac{\partial\rho_w}{\partial p}\frac{dp}{dt} = q_f - q_s$$

$$a_3\rho_w h_w\frac{dV_{wt}}{dt} + a_4m_tC_p\frac{\partial t_s}{\partial p}\frac{dp}{dt} = Q + q_f h_f - q_s h_s$$

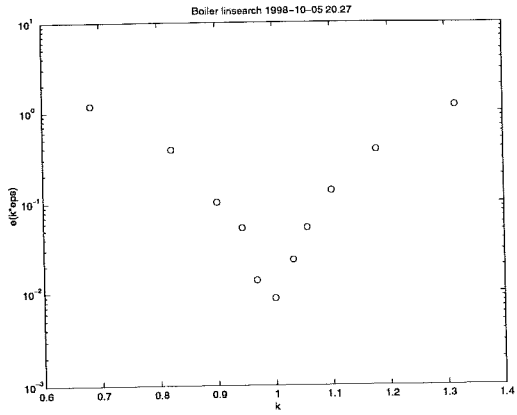

Figure 3.6 A line search along the newton direction to find a good value of ε is done as described in Section 2.6. The error ε is plotted for different values of k in (2.18).

with $a_i = 1+\varepsilon_i$ and ε given by (3.4). The model is considerably simpler than the original one (3.1). The state error caused when reducing the original model (3.1) to this model is shown in Figure 3.5 (.–.–).

3.5 Fourth order model

In this section the fourth order model is simplified in a similar way as the second order model were in Section 3.2 to Section 3.4.

Model

This section describes the fourth order model of the drum boiler as presented in Bell and Åström (1998). Two state variables are the same as for the second order model, the drum pressure p, total water volume V_{wt}. The two new states are the quality at the riser outlet α_r, and volume of steam under the liquid level in the drum V_{sd}.

The model is described on the state space form

$$
\begin{aligned}
e_{11}\frac{dV_{wt}}{dt} + e_{12}\frac{dp}{dt} &= q_f - q_s \\
e_{21}\frac{dV_{wt}}{dt} + e_{22}\frac{dp}{dt} &= Q + q_f h_f - q_s h_s \\
e_{32}\frac{dp}{dt} + e_{33}\frac{d\alpha_r}{dt} &= Q - \alpha_r h_c q_{dc} \\
e_{42}\frac{dp}{dt} + e_{43}\frac{d\alpha_r}{dt} + e_{44}\frac{dV_{sd}}{dt} &= \frac{\rho_s}{T_d}(V_{sd}^0 - V_{sd}) + \frac{h_f - h_w}{h_c} q_f,
\end{aligned}
\tag{3.5}
$$

or as

$$
0 = E(x,a)^{-1}\dot{x}(t) - B(x,u) = F(\dot{x}, x, a_1, \ldots, a_m) \tag{3.6}
$$

with

$$
E(x) = \begin{bmatrix} e_{11} & e_{12} & 0 & 0 \\ e_{21} & e_{22} & 0 & 0 \\ 0 & e_{32} & e_{33} & 0 \\ 0 & e_{42} & e_{43} & e_{44} \end{bmatrix} \qquad x = \begin{bmatrix} V_{wt} \\ p \\ \alpha_r \\ V_{sd} \end{bmatrix}
$$

where the coefficients e_{ij} are given by the following equations, with all

$a_i = 1.$

$$
\begin{aligned}
e_{11} &= a_1\rho_w - a_2\rho_s \\
e_{12} &= a_3 V_{st}\frac{\partial \rho_s}{\partial p} + a_4 V_{wt}\frac{\partial \rho_w}{\partial p} \\
e_{21} &= a_5\rho_w h_w - a_6\rho_s h_s \\
e_{22} &= a_7 V_{st}(h_s\frac{\partial \rho_s}{\partial p} + \rho_s\frac{\partial h_s}{\partial p}) + a_8 V_{wt}(h_w\frac{\partial \rho_w}{\partial p} + \rho_w\frac{\partial h_w}{\partial p}) - a_9 V_t \\
&\quad + a_{10} m_t C_p\frac{\partial t_s}{\partial p} \\
e_{32} &= a_{11}\left((1-\alpha_r)h_c\frac{\partial \rho_s}{\partial p} + \rho_s\frac{\partial h_s}{\partial p}\right)\bar{\alpha}_v V_r \\
&\quad + \left(a_{12}\rho_w\frac{\partial h_w}{\partial p} - a_{13}\alpha_r h_c\frac{\partial \rho_w}{\partial p}\right)(1-\bar{\alpha}_v)V_r \\
&\quad + a_{14}\big(\rho_s + (\rho_w - \rho_s)\alpha_r\big)h_c V_r\frac{\partial \bar{\alpha}_v}{\partial p} - a_{15}V_r + a_{16} m_r C_p\frac{\partial t_s}{\partial p} \\
e_{33} &= a_{17}\big((1-\alpha_r)\rho_s + \alpha_r\rho_w\big)h_c V_r\frac{\partial \bar{\alpha}_v}{\partial \alpha_r} \\
e_{42} &= a_{18} V_{sd}\frac{\partial \rho_s}{\partial p} \\
&\quad + \frac{1}{h_c}\Big(a_{19}\rho_s V_{sd}\frac{\partial h_s}{\partial p} + a_{20}\rho_w V_{wd}\frac{\partial h_w}{\partial p} - a_{21}V_{sd} + a_{22} m_d C_p\frac{\partial t_s}{\partial p}\Big) \\
&\quad + a_{23}\alpha_r(1+\beta)V_r\Big(\bar{\alpha}_v\frac{\partial \rho_s}{\partial p} + (1-\bar{\alpha}_v)\frac{\partial \rho_w}{\partial p} + (\rho_s - \rho_w)\frac{\partial \bar{\alpha}_v}{\partial p}\Big) \\
e_{43} &= a_{24}\alpha_r\beta(\rho_s - \rho_w)V_r\frac{\partial \bar{\alpha}_v}{\partial \alpha_r} \\
e_{44} &= a_{25}\rho_s.
\end{aligned}
\tag{3.7}
$$

where

$$\begin{aligned}
\frac{\partial \bar{\alpha}_v}{\partial p} &= \frac{1}{(\rho_w-\rho_s)^2}\Big(\rho_w\frac{\partial \rho_s}{\partial p}-\rho_s\frac{\partial \rho_w}{\partial p}\Big)\\
&\quad\Big(1+\frac{\rho_w}{\rho_s}\frac{1}{1+\eta}-\frac{1}{\eta}(1+\frac{\rho_w}{\rho_s})\ell n(1+\eta)\Big) \qquad (3.8)\\
\frac{\partial \bar{\alpha}_v}{\partial \alpha_r} &= \frac{\rho_w}{\rho_s\eta}\Big(\frac{1}{\eta}\ell n(1+\eta)-\frac{1}{1+\eta}\Big),
\end{aligned}$$

$$\eta = \alpha_r(\rho_w-\rho_s)/\rho_s$$

$$\begin{aligned}
\bar{\alpha}_v &= \frac{\rho_w}{\rho_w-\rho_s}\left[1-\frac{\rho_s}{(\rho_w-\rho_s)\alpha_r}\ell n\left(1+\frac{\rho_w-\rho_s}{\rho_s}\alpha_r\right)\right]\\
V_{wd} &= V_{wt}-V_{dc}-(1-\bar{\alpha}_v)V_r\\
\ell &= \frac{V_{wd}+V_{sd}}{A_d}\\
T_d &= \frac{\rho_s V_{sd}^0}{q_{sd}}\\
kq_{dc}^2 &= 2\rho_w A_{dc}(\rho_w-\rho_s)g\bar{\alpha}_v V_r\\
q_r &= q_{dc}-V_r\Big(\bar{\alpha}_v\frac{\partial \rho_s}{\partial p}+(1-\bar{\alpha}_v)\frac{\partial \rho_w}{\partial p}+(\rho_w-\rho_s)\frac{\partial \bar{\alpha}_v}{\partial p}\Big)\frac{dp}{dt}\\
&\quad+(\rho_w-\rho_s)V_r\frac{\partial \bar{\alpha}_v}{\partial \alpha_r}\frac{d\alpha_r}{dt}.
\end{aligned}$$

The same inputs as for the second order model are used, shown in Figure 3.2. A simulation of the model is shown in Figure 3.7 (—).

Choice of components to reduce

For larger models it can be difficult to know which parameters that can be neglected without to much loss of accuracy of the model. The calculations done in Section 2.6 can be used not only to adjust remaining parameters, but also to choose which parameters to neglect. Then different coefficients a_{ij} are put in front of each parameter, like

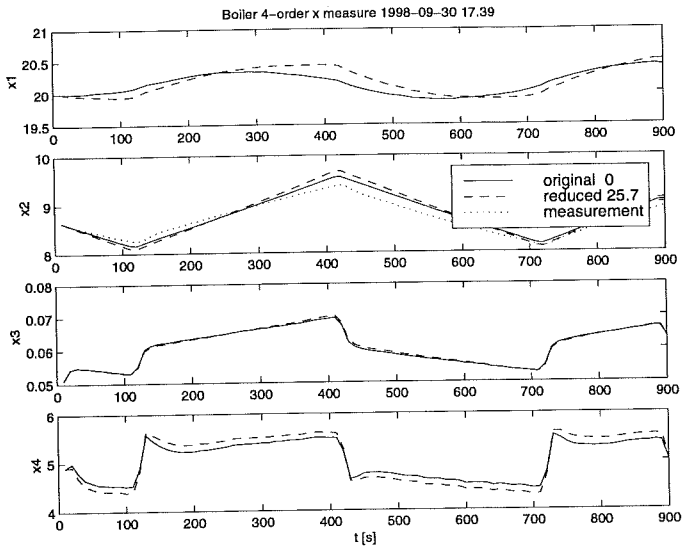

Figure 3.7 Simulation of the boiler model (3.5) and the reduced model with the input signals shown in Figure 3.2. The state error e in (2.7) is written in the box. The second state $x_2 = p$ is compared with measurements.

in (3.7). No coefficient is denoted a_0 since it is not yet known what to reduce. Simulations of $d_k(t)$ and c_{ij} will give a C from (2.14) with the following first column

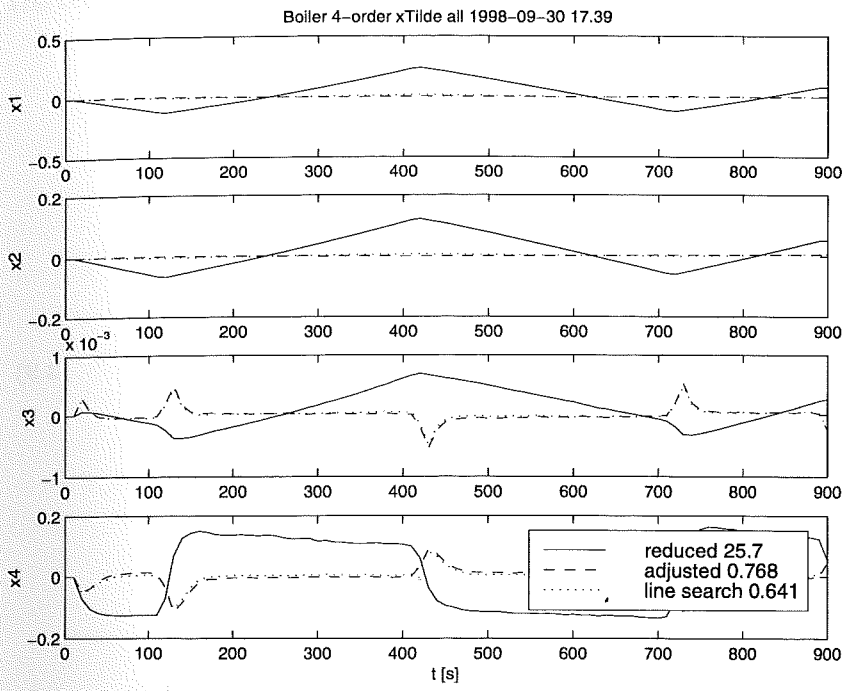

Figure 3.8 Simulation of the boiler model (3.5). The state error $\tilde{x}_i$ is compared for the the reduced model, the compensated model, and the model compensated with the result of the line search, all with the input signals shown in Figure 3.2. The norm of the state error e in (2.7) is written in the box for the different cases.

$$C_0 = \begin{bmatrix} c_{01} \\ c_{02} \\ \hline c_{03} \\ c_{04} \\ \hline c_{05} \\ c_{06} \\ \hline c_{07} \\ c_{08} \\ c_{09} \\ c_{010} \\ \hline c_{011} \\ c_{012} \\ c_{013} \\ c_{014} \\ c_{015} \\ c_{016} \\ \hline c_{017} \\ \hline c_{018} \\ c_{019} \\ c_{020} \\ c_{021} \\ c_{022} \\ c_{023} \\ \hline c_{024} \\ \hline c_{025} \end{bmatrix} = 10^8 \begin{bmatrix} 0.1450 \\ \mathbf{0.5867} \\ \hline 0.4557 \\ \mathbf{-1.1782} \\ \hline \mathbf{-0.7060} \\ 0.0981 \\ \hline 0.0729 \\ 0.0374 \\ -0.0077 \\ \mathbf{0.4938} \\ \hline 0.0156 \\ \mathbf{0.1544} \\ 0.0072 \\ \mathbf{-0.0256} \\ -0.0063 \\ \mathbf{0.0819} \\ \hline \mathbf{-0.2309} \\ \hline 0.0140 \\ -0.0013 \\ \mathbf{-0.1652} \\ -0.0014 \\ \mathbf{0.0270} \\ 0.0123 \\ \hline \mathbf{0.2244} \\ \hline \mathbf{-0.1045} \end{bmatrix} \tag{3.9}$$

The magnitude of each c_{0j} value indicates how much the error e in (2.7) will increase when a certain parameter associated with a_j is neglected. Discussions with the authors of the drum boiler model Åström and Bell (1998), combined with studies of the c_{0j} factors have determined

which parts of the model to neglect. The terms to neglect are chosen so that in each e_{ij} coefficient at least one term will be kept. The parts that are chosen to keep are the a_j:s in (3.7) corresponding to a c_{oj} written with bold face in (3.9). Therefore all a_i corresponding to a non bold face c_{0j} will be replaced with a_0 and the other a_i:s will be renumbered. The reduced model coefficients are then

$$
\begin{aligned}
e_{11} &= a_1\rho_w \\
e_{12} &= a_2 V_{wt}\frac{\partial \rho_w}{\partial p} \\
e_{21} &= a_3\rho_w h_w \\
e_{22} &= a_4 m_t C_p \frac{\partial t_s}{\partial p} \\
e_{32} &= -a_5\alpha_r h_c \frac{\partial \rho_w}{\partial p}(1-\bar{\alpha}_v)V_r \\
&\quad + a_6(\rho_w-\rho_s)\alpha_r)h_c V_r \frac{\partial \bar{\alpha}_v}{\partial p} + a_7 m_r C_p \frac{\partial t_s}{\partial p} \\
e_{33} &= a_8\big((1-\alpha_r)\rho_s + \alpha_r\rho_w\big)h_c V_r \frac{\partial \bar{\alpha}_v}{\partial \alpha_r} \\
e_{42} &= \frac{1}{h_c}\Big(a_9\rho_w V_{wd}\frac{\partial h_w}{\partial p} + a_{10} m_d C_p \frac{\partial t_s}{\partial p}\Big) \\
e_{43} &= a_{11}\alpha_r\beta(\rho_s-\rho_w)V_r\frac{\partial \bar{\alpha}_v}{\partial \alpha_r} \\
e_{44} &= a_{12}\rho_s.
\end{aligned}
\tag{3.10}
$$

(3.9)

in (2.7)
glected.
m and
rmined

Adjustment

The adjustment procedure described in Section 2.6 is applied. The state error for the reduced and compensated model is shown in Figure 3.8. A line search gives the result shown in Figure 3.9. The best value of

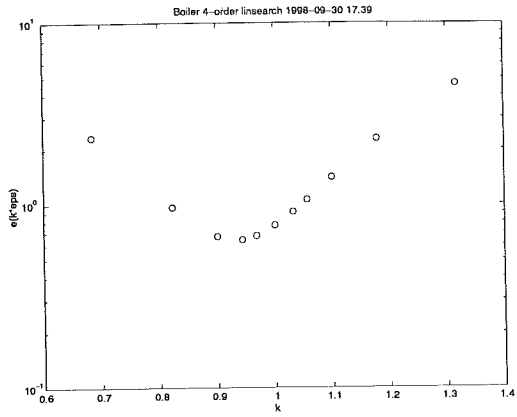

Figure 3.9 A line search is done along the newton direction to find a good value of ε. The error ε is plotted for different values of k in (2.18).

k seem to be $k = 0.94$. This results in the following ε value.

$$\varepsilon = \begin{bmatrix} 0.0572 \\ 0.3273 \\ 0.1249 \\ -0.1996 \\ 0.1314 \\ 1.5085 \\ 0.0007 \\ 0.2562 \\ -0.0418 \\ -1.0789 \\ 0.2741 \\ -0.0117 \end{bmatrix} \tag{3.11}$$

Results

After reduction, adjustment and line search the model is described by (3.5) with (3.10) and $a_1 = 1 + \varepsilon_1, \ldots, a_m = 1 + \varepsilon_m$ with ε from (3.11). The state error is shown in Figure 3.8 (...). A disadvantage with adjusting parameters is that some physical insight might get lost, but the error of the states is reduced by more than one order of magnitude.

good

(3.11)

4

Error Bounds

4.1 Introduction

In Chapter 2 it was discussed how to minimize the error when reducing a nonlinear model. In this chapter it is shown that in some cases a strict upper bound of the error can be found. The problem is formulated in Section 4.2 and solved for the scalar case in Section 4.3 and the vector case Section 4.4. The pendulum example from Section 2.3 is treated in Section 4.5.

4.2 Problem formulation

In order to keep the formulas simple, the notation in this chapter is not as general as in Chapter 2. The ideas are developed for models in explicit form

$$\dot{x}(t) = f(x(t))$$

but could be extended to models in implicit form (2.1). The notation introduced is analogous to (2.1) ... (2.4). Suppose that a model contains a number of parameters $a_0, \ldots, a_m$ that can be changed or neglected. The model is then written as

$$\dot{x}(t) = f(x(t), a) = f(x(t), a_0, \ldots, a_m) \tag{4.1}$$

with

$$a = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}$$

The nominal model is now written as

$$\dot{x}(t) = f(x(t), 1, \ldots, 1) \tag{4.2}$$

where the $a_i = 1$ corresponds to the original model. Suppose further that there is a hypothesis that some parts of the model can be omitted. The reduced model is written as

$$\dot{\bar{x}}(t) = f(\bar{x}(t), 0, 1, \ldots, 1) \tag{4.3}$$

with

$$\bar{a} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

The goal now it to find an upper bound on the state error introduced by the reduction.

$$\tilde{x}(t) = x(t) - \bar{x}(t)$$

Norms

The signal norm used in this chapter is a type of $\|\cdot\|_\infty$ norm, extended to handle a time index t as well as a state index i. Normal $\|\cdot\|_\infty$ norms are described in Doyle *et al.* (1992).

$$\|x(t)\| = \max_{\substack{t \\ i}} |x_i(t)| \tag{4.4}$$

This definition of vector norm induces the operator norm

$$\|A(t)\| = \max_{x(t)\neq 0} \frac{\|A(t)x(t)\|}{\|x(t)\|}$$

that can be calculated by

$$\|A(t)\| = \max_i \sum_j \max_t a_{ij}(t) \tag{4.5}$$

4.3 Scalar case

The method of finding error bounds is described via a small scalar example.

Example

Consider the system

$$\dot{x} = 0.1\frac{\sin x}{2} + \frac{\sin 2x}{2} + 1 \tag{4.6}$$

Suppose that the term $\frac{\sin x}{2}$ can be omitted. If the model is written as

$$\dot{x} = f(x,a) = 0.1a_0\frac{\sin x}{2} + a_1\frac{\sin 2x}{2} + a_2 \tag{4.7}$$

the nominal model is given by (4.6) and the reduced is written as

$$\dot{\bar{x}} = f(\bar{x},\bar{a}) = \frac{\sin 2\bar{x}}{2} + 1 \tag{4.8}$$

A simulation of the original system and the reduced system is shown in Figure 4.1.

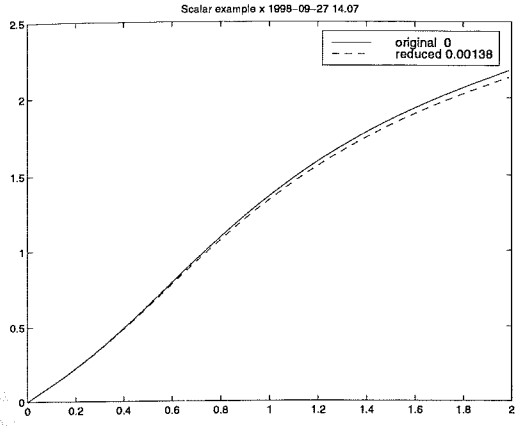

Figure 4.1 Simulation of the original system (4.6) and the reduced system (4.8). The norm of the state error, e defined in (2.7), is written in the box.

Estimation of error

The differential equation approximating the error $\tilde{x}$ is a special case of (2.12).

$$\begin{aligned}\dot{\tilde{x}} &= f(x,a) - f(\bar{x},\bar{a}) = \\ &= f(x,\bar{a}) - f(\bar{x},\bar{a}) + f(x,a) - f(x,\bar{a}) = \\ &\frac{\partial f(\bar{x},\bar{a})}{\partial x}\tilde{x} + \xi + f(x,a) - f(x,\bar{a}) \\ &\approx \frac{\partial f(\bar{x},\bar{a})}{\partial x}\tilde{x} + f(x,a) - f(x,\bar{a})\end{aligned} \tag{4.9}$$

where ξ is introduced to represent the error introduced by the approximations.

$$\xi = f(x,\bar{a}) - f(\bar{x},\bar{a}) - \frac{\partial f(\bar{x},\bar{a})}{\partial x}\tilde{x} \tag{4.10}$$

The error $\tilde{x}$ is can now be simulated with (4.9).

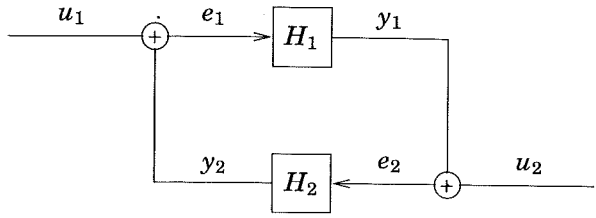

Figure 4.2 Illustration of the Small Gain Theorem (4.11).

Small Gain Theorem

The goal in this section is to find an upper bound on the error, estimated in the previous section. This is achieved using the Small Gain Theorem, in Khalil (1996) stated as follows.

Consider the system shown in Figure 4.2. Suppose H_1 and H_2 are stable systems with the gains γ_1 and γ_2, i.e.,

$$\|y_1\| \leq \gamma_1\|e_1\| + \beta_1$$
$$\|y_2\| \leq \gamma_2\|e_2\| + \beta_2$$

Then

$$\|e_1\| \leq \frac{1}{1-\gamma_1\gamma_2}\left(\|u_1\| + \gamma_2\|u_2\| + \beta_1 + \beta_2\right) \tag{4.11}$$

Figure 4.2 shows the Small Gain Theorem is applied to (4.9), resulting in an upper bound on the gain

$$\frac{\|\tilde{x}\|}{\|f(x,a) - f(x,\bar{a})\|} \leq \frac{\|G\|}{1 - \|G\|\|L\|} \tag{4.12}$$

where L is the operator defined by the relation

$$\xi \leq L\tilde{x} \tag{4.13}$$

and G is the system

$$\dot{\tilde{x}} = \frac{\partial f(\bar{x},\bar{a})}{\partial x}\tilde{x} + u \tag{4.14}$$

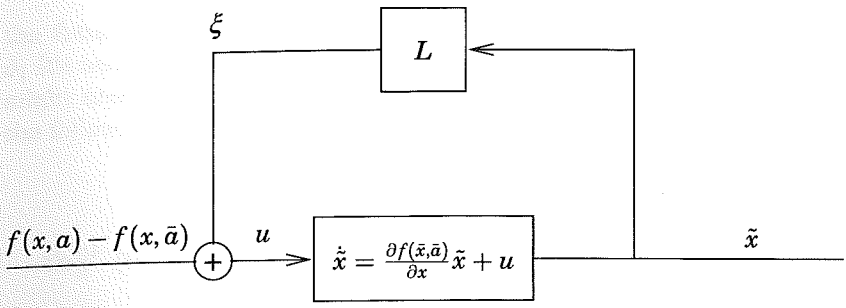

Figure 4.3 The Small Gain Theorem (4.11) applied to (4.9) gives an upper bound on the gain of the system, as shown in (4.12).

If bounds on $\|L\|$, $\|f(\bar{x}, a) - f(\bar{x}, \bar{a})\|$ and $\|G\|$ can be found, an upper bound on $\tilde{x}(t)$ can be calculated from (4.12). The bound are found in the following subsections.

System gain

The gain of a linear time-varying system G can be found via the transition matrix $\Phi(t, 0)$, see Rugh (1996). When the input to a system is bounded $\|u(t)\| \leq u_{max}$ the output $y(t)$ can be written as

$$\|y(t)\| = \left\| \int_0^t \Phi(t, \tau) u(\tau) d\tau \right\| \leq u_{max} \int_0^t \left\| \Phi(t, 0) \Phi(\tau, 0)^{-1} \right\| d\tau \tag{4.12}$$

giving

$$\|G\| \leq \int_0^t \left\| \Phi(t, 0) \Phi(\tau, 0)^{-1} \right\| d\tau \tag{4.15}$$

This integral can be evaluated for some discrete values of t. A continuous curve describing the maximum gain can be found with the

following calculations

$$\|y(t)\| = \left\| \int_0^t \Phi(t,\tau)u(\tau)d\tau \right\| \leq \int_0^t \|\Phi(t,\tau)u(\tau)\|\, d\tau \leq$$

$$\int_0^t \|\Phi(t,0)\| \cdot \left\|\Phi(\tau,0)^{-1}\right\| \|u_{max}\|\, d\tau \leq$$

$$\leq u_{max} \|\Phi(t,0)\| \int_0^t \left\|\Phi(\tau,0)^{-1}\right\| d\tau$$

$$\|G\| \leq \|\Phi(t,0)\| \int_0^t \left\|\Phi(\tau,0)^{-1}\right\| d\tau \tag{4.16}$$

For the vector case (4.15) give a more tight bound on the gain then (4.16) but for the scalar case they are equally exact.

Gain of L

An expression for the gain L is found when using Taylors Theorem, stated in Rudin (1976) as follows.

Suppose $f(t)$ is real, $f^{(n-1)}(t)$ is continuous and $f^{(n)}(t)$ exist on $[a,b]$ for a positive integer n. Let α and β be distinct points of $[a,b]$. Then there exist an $\alpha < x < \beta$ such that

$$f(\beta) = \sum_{k=0}^{n-1} \frac{f^k(\alpha)}{k!}(\beta-\alpha)^k + \frac{f^n(x)}{n!}(\beta-\alpha)^k \tag{4.17}$$

Applying this with $n = 2$ to (4.10) gives that there exist a $\theta \in [0,1]$ such that

$$f(x,\bar{a}) = f(\bar{x},\bar{a}) + \frac{\partial f(\bar{x},\bar{a})}{\partial x}\tilde{x} + \underbrace{\underbrace{\frac{1}{2}\frac{\partial^2 f(\bar{x}+\theta\tilde{x},\bar{a})}{\partial x^2}}_{L}\tilde{x}\,\tilde{x}}_{\xi} \tag{4.18}$$

An upper bound on $\|L\|$ is needed when $\|\xi\| \leq \|L\|\|\tilde{x}\|$. This will be easier to find if a maximum state error $\|\tilde{x}(t)\| \leq \delta_1$ is introduced. If

the future calculations result in a maximum state error larger than δ_1 it's necessary to start from this point again with a another δ_1. This might need a few iterations. To find an upper bound on the gain of L, the following expression needs to be simplified.

$$\|L\| = \max_{\substack{t\in[0,T]\\ \theta\in[0,1]\\ \|\tilde{x}(t)\|\in[0,\tilde{x}_{max}]}} \left\| \frac{1}{2}\frac{\partial^2 f(\bar{x}+\theta\tilde{x},\bar{a})}{\partial x^2}\tilde{x} \right\| \tag{4.19}$$

Different methods might be used depending on the structure of L in different cases.

In the discussion above the bound on L is used to find a bound on $\tilde{x}(t)$ using the Small Gain Theorem. On the other hand the bound on $\tilde{x}(t)$ is used to find a bound on L since the expression for L in (4.18) include $\tilde{x}(t)$. To avoid a circle argument, it is used that the error is zero from the beginning. The Grönwall-Bellman inequality Khalil (1996) is applied as follows.

1. The initial conditions for the original model (4.2) and the reduced model (4.3) are equal. Therefor $\tilde{x}(t_0) = 0$.
2. Suppose that the function f is Lipschitz with respect to x. Then the Grönwall-Bellman inequality states that $x(t)$ and $\bar{x}(t)$ and therefor also $\tilde{x}(t)$ can grow at the most exponentially, see Figure 4.4. Then for any number δ_1 it's known ,that $\tilde{x}(t) \leq \delta_1$ for at least some short time interval $t \in [t_0, t_1]$.
3. Now there exist a bound on $\|\tilde{x}(t)\| \leq \delta_1$ for the time interval. This can be used in (4.19) to find an upper bound on L.
4. The Small Gain Theorem can then be applied for the specific time interval $t \in [t_0, t_1]$ giving a bound on $\tilde{x}(t)$, that is called δ_2. If $\delta_2 \geq \delta_1$ one has to start over from point 2 with a another δ_1. If however $\delta_2 \leq \delta_1$ it follows that not only $\|\tilde{x}(t)\| \leq \delta_1$ but also $\|\tilde{x}(t)\| \leq \delta_2$.
5. Knowing that $\tilde{x}(t_1) \leq \delta_2$ Grönwall-Bellman inequality can be used again to show that for some short time interval $t \in [t_1, t_2]$ $\tilde{x}(t) \leq \delta_1$ since $\tilde{x}(t)$ grows at the most exponentially. Point three and four can now be repeated for the time interval $t \in [t_1, t_2]$.

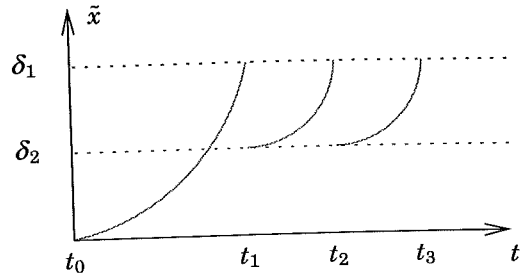

Figure 4.4 Illustration of the iterative procedure described on the previous page to find strict error bounds on L and on $\tilde{x}(t)$.

6. Point two to five can be repeated until it is shown that $\|\tilde{x}(t)\| \leq \delta_2$ for all t. The time intervals $[t_1, t_2]$, $[t_2, t_3]$, ... will be equally large.

Example continued

The example introduced in the beginning of Section 4.3 is now discussed further. The state error caused when the system (4.6) is reduced to (4.8) is simulated with (4.9). The result of this simulation is shown in Figure 4.5 and compared with the actual error, i.e., the difference between the trajectories (—) and (- -) in Figure 4.1. The fact that the curves are close to each other, shows that the error ξ in the estimation of $\tilde{x}$ in (4.9) is small. Figure 4.6 shows all the components of f from a simulation of (4.6).

The transition matrix $\Phi(t, 0)$ for the system (4.14) is shown in Figure 4.7. It is discussed in Rugh (1996) how to calculate a transition matrix. Further the system gain $\|G\|$ of the system (4.14) is calculated according to (4.15) as well as (4.16), both shown in Figure 4.8. The plot gives

$$\|G\| \leq 1.2$$

In order to use Taylors formula like in (4.18), the Jacobian and Hessian of the system (4.8) is calculated.

$$\frac{\partial f(x, \bar{a})}{\partial x} = \cos 2x$$

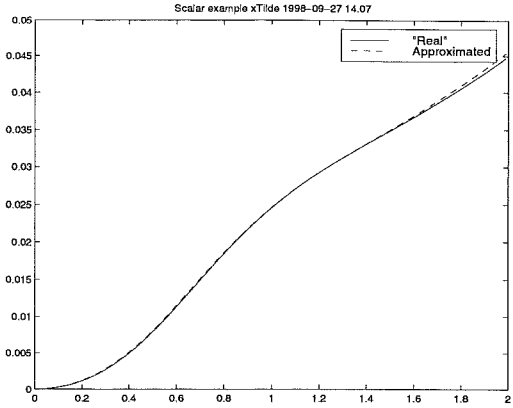

Figure 4.5 The state error $\tilde{x}$ caused by reducing the model (4.6) to (4.8). An approximation of the state error (- -) simulated with (4.9) is compared with the actual difference (—) between the simulations of (4.6) and (4.8), both shown in Figure 4.1.

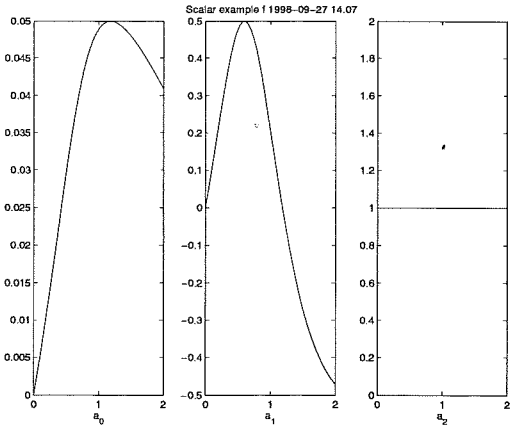

Figure 4.6 The different components of f in (4.7). The difference $f(x,a) - f(x,\tilde{a})$ is shown in the left plot and the norm $\|f(x,a) - f(x,\tilde{a})\| \leq 0.05$. This will be needed to use the Small Gain Theorem as in (4.12).

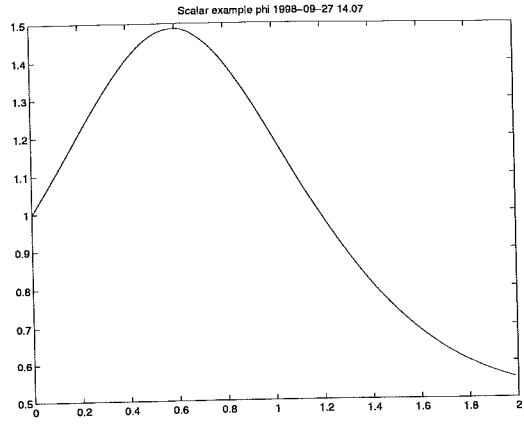

Figure 4.7 The transition matrix $\Phi(t,0)$ for the system (4.6).

$$\frac{\partial^2 f(x,\bar{a})}{\partial x^2} = -2\sin 2x$$

Some approximations of the gain of L in (4.19) is done

$$\|L\| = \max_{\substack{t\in[0,T]\\ \theta\in[0,1]\\ \|\tilde{x}(t)\|\in[0,\tilde{x}_{max}]}} \left\| \frac{1}{2}\frac{\partial^2 f(\bar{x}+\theta\tilde{x},\bar{a})}{\partial x^2}\tilde{x} \right\| =$$

$$= \frac{1}{2}\max \|-2\sin(2(\bar{x}+\theta\tilde{x}))\|\,\tilde{x}_{max} \le \frac{1}{2}\,\|-2\|\,\tilde{x}_{max} = \tilde{x}_{max}$$

The estimation

$$\left\| \frac{\partial^2 f(\bar{x}+\theta\tilde{x},\bar{a})}{\partial x^2} \right\| \le 2 \tag{4.20}$$

corresponds well to Figure 4.9 where

$$\max_{t\in[0,T]} \left\| \frac{\partial^2 f(\bar{x},\bar{a})}{\partial x^2} \right\| = 2 \tag{4.21}$$

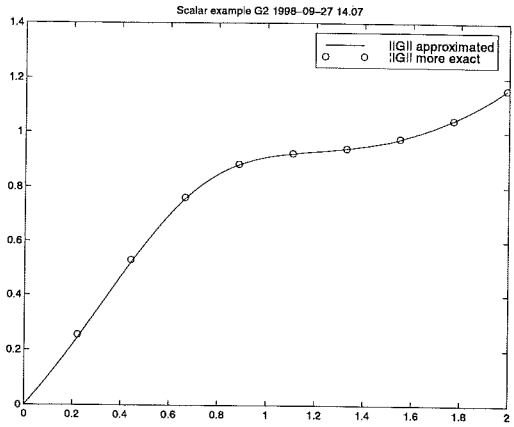

Figure 4.8 The maximum gain $\|G\|$ of the system (4.14) for the scalar example (4.6). The gain is calculated with (4.15) (o) as well as with (4.16) (—). For this scalar example the two formulas give the same result.

It is now needed to choose a value of δ_1 to use as a hypothesis. Since the estimated $\tilde{x} \leq 0.05$ in Figure 4.5 for the considered time interval, a natural choice is $\delta_1 = 0.05$. The application of the Small Gain Theorem in (4.12) gives

$$\frac{\|\tilde{x}\|}{\|f(\bar{x},a) - f(\bar{x},\bar{a})\|} \leq \frac{\|G\|}{1 - \|G\|\|L\|} \leq$$

$$\leq \frac{\|1.2\|}{1 - \|1.2\|\tilde{x}_{max}} \leq \frac{1.2}{1 - 1.2 \cdot 0.05} \leq 1.3 \quad (4.22)$$

Figure 4.6 gives

$$\| - f(x,a) + f(x,\bar{a})\| \leq 0.05 \quad \Longrightarrow \quad \|\tilde{x}\| \leq 1.3 \cdot 0.05 = 0.065$$

Since $\delta_2 = 0.065 > 0.05 = \delta_1$ the hypothesis $\tilde{x}_{max} \leq 0.05$ can not be verified. A possible second try is $\delta_1 = 0.1$. The calculations above are repeated.

$$\frac{\|\tilde{x}\|}{\| - f(\bar{x},a) + f(\bar{x},\bar{a})\|} \leq \frac{\|G\|}{1 - \|G\|\|L\|} \leq \frac{1.2}{1 - 1.2 \cdot 0.1} \leq 1.5$$

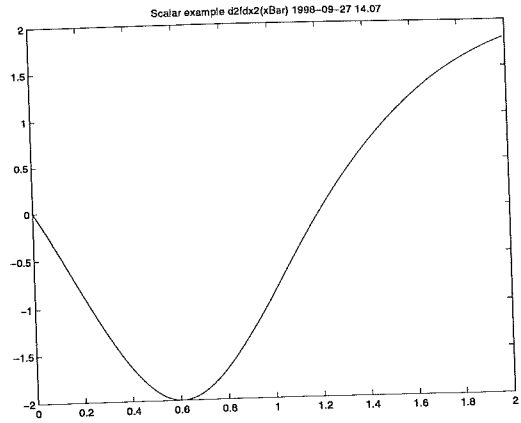

Figure 4.9 The Hessian $\frac{\partial^2 f(\bar{x},\bar{a})}{\partial x^2}$ of the system (4.6) evaluated for the trajectory $\bar{x}(t)$ simulated with (4.8).

$$\| - f(\bar{x}, a) + f(\bar{x}, \bar{a})\| \leq 0.05 \longrightarrow \|\tilde{x}\| \leq 0.075$$

Since $\delta_2 = 0.075 < 0.1 = \delta_1$ the hypothesis $\tilde{x}_{max} \leq 0.10$ can be verified.

Conclusion

An upper bound on the error $\tilde{x}(t)$ caused by setting a_0 equal to zero in (4.7) is found.

$$\|\tilde{x}\| \leq 0.075$$

4.4 Vector case

In this section the derivations of Section 4.3 is generalized to the vector case, i.e systems of order higher than one. Derivations that are different from the scalar case are carried out. The ideas are exemplified with a small second order example.

The following matrix notation is introduced

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{bmatrix} \qquad \xi = \begin{bmatrix} \xi_1 \\ \vdots \\ \xi_n \end{bmatrix}$$

Equation (4.9), (4.10) and (4.12) then can be written identically to the scalar case and Taylors formula (4.18) can be generalized to

$$f_k(x,\bar{a}) = f_k(\bar{x},\bar{a}) + \frac{\partial f_k(\bar{x},\bar{a})}{\partial x}\tilde{x} + \underbrace{\frac{1}{2}\tilde{x}^T \frac{\partial^2 f_k(\bar{x}+\theta_k\tilde{x},\bar{a})}{\partial x^2}\tilde{x}}_{\xi_k} \quad k \in \{1,2,\dots,n\} \tag{4.23}$$

where

$$\theta_k = diag(\theta_{k1},\dots,\theta_{kn}), \quad \theta_{ki} \in [0,1]$$

$$\frac{\partial f_k}{\partial x} = \begin{bmatrix} \frac{\partial f_k}{\partial x_1} & \cdots & \frac{\partial f_k}{\partial x_n} \end{bmatrix}$$

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}$$

$$\frac{\partial^2 f_k}{\partial x^2} = \begin{bmatrix} \frac{\partial^2 f_k}{\partial x_1^2} & \cdots & \frac{\partial^2 f_k}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f_k}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f_k}{\partial x_n^2} \end{bmatrix}$$

This notation will become clear in the example that follows. Each equation (4.23) can be treated analogous to what is done to (4.18) in (4.13) and (4.19). This can be written with matrix notation as

$$\xi \leq L\tilde{x}$$

$$\|L\| = \max_{\substack{t\in[0,T] \\ \theta_k\in[0,1] \\ \|\tilde{x}(t)\|\in[0,\tilde{x}_{max}]}} \left\| \begin{matrix} \frac{1}{2}\tilde{x}^T \frac{\partial^2 f_1(\bar{x}+\theta_1\tilde{x},a)}{\partial x^2} \\ \vdots \\ \frac{1}{2}\tilde{x}^T \frac{\partial^2 f_n(\bar{x}+\theta_n\tilde{x},a)}{\partial x^2} \end{matrix} \right\| \tag{4.24}$$

This expression needs to be simplified, with methods that can be specific for each problem. The calculations (4.15) and (4.16) are exactly identical in the vector case.

Example

In this section error bounds are found for the following system.

$$f(x) = \begin{bmatrix} 0.2\sin(x_1) + \sin(x_2) \\ \cos(x_1) \end{bmatrix} \tag{4.25}$$

The hypothesis is that the $0.2\sin(x_1)$ component is negligible. Hence the system (4.25) is written as

$$f(x,a) = \begin{bmatrix} 0.2a_0\sin(x_1) + a_1\sin(x_2) \\ a_2\cos(x_1) \end{bmatrix} \tag{4.26}$$

and the reduced system as

$$f(\bar{x},\bar{a}) = \begin{bmatrix} \sin(\bar{x}_2) \\ \cos(\bar{x}_1) \end{bmatrix}. \tag{4.27}$$

A simulation of the nominal system (4.25) compared with the reduced system (4.27) is shown in Figure 4.10. A simulation of the state error from (4.9) is shown in Figure 4.11. The Jacobian and Hessian are needed for the Taylor expansion (4.23).

$$\frac{\partial f(x,\bar{a})}{\partial x} \begin{bmatrix} 0 & \cos(x_2) \\ -\sin(x_1) & 0 \end{bmatrix}$$

$$\frac{\partial^2 f_1(x,\bar{a})}{\partial x^2} = \begin{bmatrix} 0 & 0 \\ 0 & -\sin(x_2) \end{bmatrix}$$

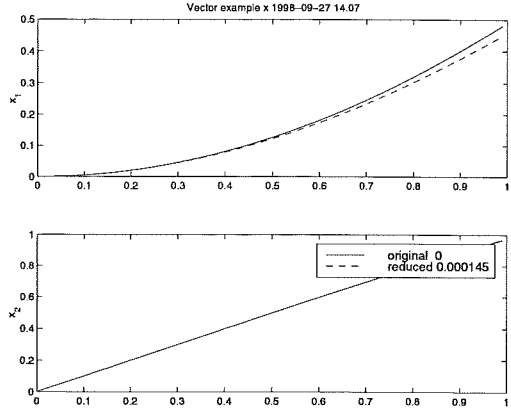

Figure 4.10 Simulation of the original system (4.25) and the reduced system (4.27). The norm of the state error, e defined in (2.7), is written in the box.

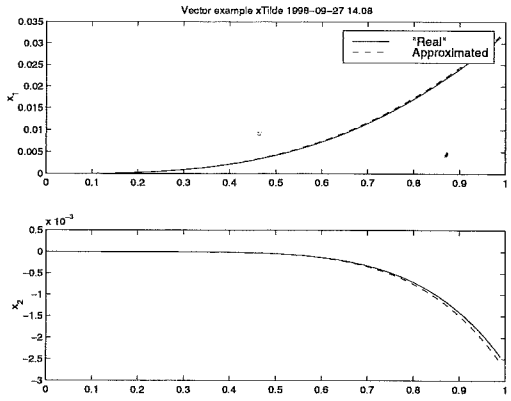

Figure 4.11 The state error $\tilde{x}$ caused by reducing the model (4.25) to (4.27). An approximation of the state error (- -) simulated with (4.9) is compared with the actual difference (—) between the simulations of (4.25) and (4.27), both shown in Figure 4.10.

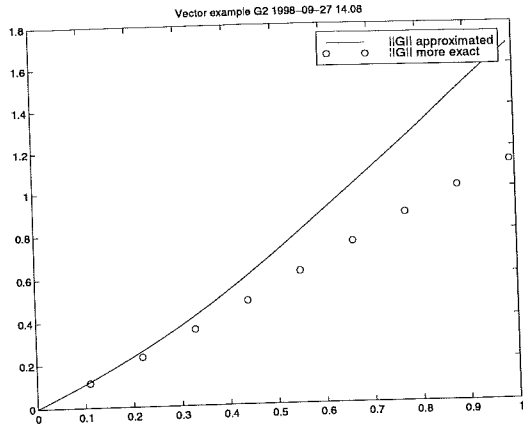

Figure 4.12 The maximum gain $\|G\|$ of the system (4.14) for the vector example (4.25). The gain is calculated with (4.15) (o) as well as with (4.16) (—). For this vector example (4.16) gives a more conservative approximation than (4.15).

$$\frac{\partial^2 f_2(x,\bar{a})}{\partial x^2} = \begin{bmatrix} -\cos(x_1) & 0 \\ 0 & 0 \end{bmatrix}$$

Some approximations of the gain L in (4.24) for the system (4.25) can be made.

$$L = \max \left\| \begin{matrix} \frac{1}{2}\tilde{x}^T \frac{\partial^2 f_1(\bar{x}+\theta_1\tilde{x},a)}{\partial x^2} \\ \vdots \\ \frac{1}{2}\tilde{x}^T \frac{\partial^2 f_n(\bar{x}+\theta_n\tilde{x},a)}{\partial x^2} \end{matrix} \right\| \leq \frac{1}{2}\tilde{x}_{max} \max \sum_{k=1}^{n} \left\| \frac{\partial^2 f_k(\bar{x}+\theta\tilde{x},a)}{\partial x^2} \right\|$$

$$\leq \frac{1}{2}\tilde{x}_{max} \left(\max \left\| \begin{matrix} 0 & 0 \\ 0 & -sin(x_2+\theta_1\tilde{x}_2) \end{matrix} \right\| + \max \left\| \begin{matrix} -\cos(x_1+\theta_2\tilde{x}_1) & 0 \\ 0 & 0 \end{matrix} \right\| \right)$$

$$\leq \frac{1}{2}\tilde{x}_{max}(1+1) = \tilde{x}_{max} \quad (4.28)$$

The gain G of the system (4.14) is calculated both with (4.15) and (4.16) and plotted in Figure 4.12. The (4.15) points show that

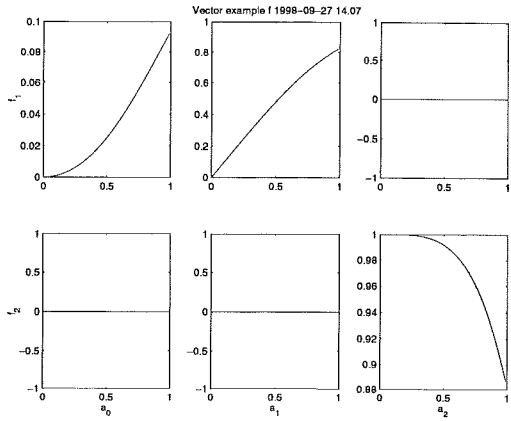

Figure 4.13 The different components of f in (4.26). The difference $f(x,a)-f(x,\bar{a})$ is shown in the left plot and the norm $\|f(x,a)-f(x,\bar{a})\| \leq 0.1$.

$$\|G\| \leq 1.2$$

In Figure 4.11 $\tilde{x} < 0.05$ why starting value for δ_1 could be $\delta_1 = 0.05$.

$$\frac{\|\tilde{x}\|}{\|-f(\bar{x},a)+f(\bar{x},\bar{a})\|} \leq \frac{\|G\|}{1-\|G\|\|L\|} \leq \frac{\|1.2\|}{1-\|1.2\|\|0.05\|} \leq 1.3$$

The different components of f in (4.25) are plotted in Figure 4.13. The upper left plot gives gives

$$\|-f(\bar{x},a)+f(\bar{x},\bar{a})\| \leq 0.1 \qquad \Longrightarrow \qquad \|\tilde{x}\| \leq 0.13$$

Since $\delta_2 = 0.13 > 0.05 = \delta_1$ the hypothesis $\|\tilde{x}\| \leq 0.05$ can not be verified. A few iterations give that a bound rather close to the best possible that can be found with this method is $\tilde{x}_{max} = 0.15$

$$\frac{\|\tilde{x}\|}{\|-f(\bar{x},a)+f(\bar{x},\bar{a})\|} \leq \frac{\|G\|}{1-\|G\|\|H\|} \leq \frac{\|1.2\|}{1-\|1.2\|\|0.15\|} \leq 1.47$$

$$\| - f(\bar{x}, a) + f(\bar{x}, \bar{a})\| \leq 0.1 \qquad \Longrightarrow \qquad \|\tilde{x}\| \leq 0.147$$

Since $\delta_2 = 0.147 < 0.15 = \delta_1$ the hypothesis $\|\tilde{x}\| \leq 0.15$ can be verified and it is concluded that $\tilde{x}_{max} < 0.15$.

Conclusion

The calculations done for the scalar case in Section 4.3 are extended with matrix notation to higher order systems. The notation and methods are exemplified with a second order example (4.25). An upper bound on the error $\tilde{x}(t)$ caused by setting a_0 equal in (4.26) is found.

$$\|\tilde{x}\| \leq 0.147$$

4.5 Pendulum example

Now the pendulum example presented in Section 2.3 and Section 2.7 is treated again. The purpose is to find an upper bound on the error, when the rotating pendulum is approximated by a non-rotating one. Both models are in (2.10) written with the notation (4.1) ... (4.3).

$$\dot{x} = f(x, a) = \begin{bmatrix} a_3 x_2 \\ a_0 \frac{x_3^2}{2} \sin 2x_1 + a_1 \frac{g}{L_2} \sin x_1 - a_2 \frac{u}{L_2} \cos x_1 \\ a_4 \frac{u}{L_1} \end{bmatrix} \tag{4.29}$$

The Jacobian is identical to (2.20) except for the x_3^2-term but Hessian of the system is also needed for the error bound calculations. The Hessian will involve quite complex expressions containing second derivatives of the control law (2.19) why an approximation of L like (4.28) will be complicated and lead to high values of $\|L\|$.

Control law saved

To avoid the problems described above a stored control signal is used. The control signal (2.19) is used in the simulations in Chapter 2 and then stored. The simulation in this chapter uses the stored signal, with no feedback, i.e., open loop control is used. Then the pendulum can not be balanced in an upwards position. The time interval studied include

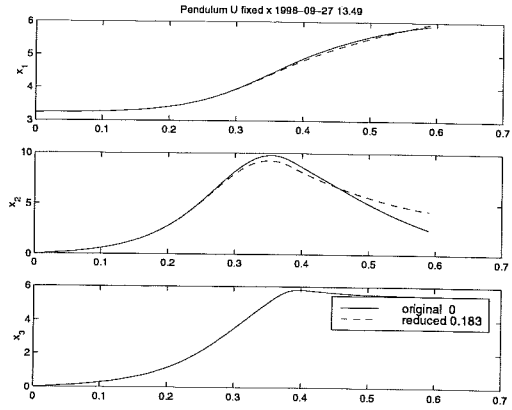

Figure 4.14 Simulation of the original system (4.29) with all $a_i = 1$ and the reduced system where $a_0 = 0$. The norm of the state error, e defined in (2.7), is written in the box.

only the swing up, not the balancing. The control signal identical to the first 0.7 second of Figure 2.5 and the resulting trajectory is shown in Figure 4.14. The state error $\tilde{x}$ are simulated using (4.9) and in Figure 4.15 compared with the difference between the two simulations in Figure 4.14.

Jacobian and Hessian

Since the control signal u is independent of the states and the the reduced model (4.29) with $a_0 = 0$ is used, the Jacobian become rather simple.

$$\frac{\partial f(x,\bar{a})}{\partial x} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{g}{L_2}\cos x_1 + \frac{u}{L_2}\sin x_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

This leads to a Hessian where all of the 27 elements except one is equal to zero

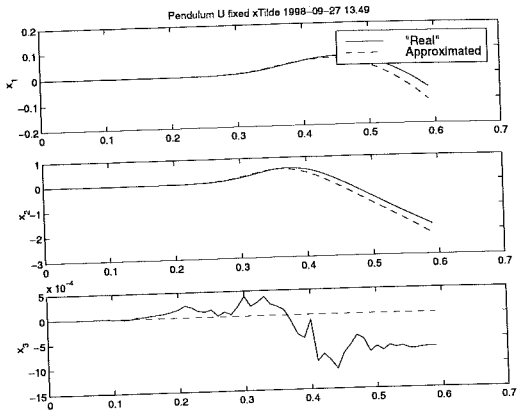

Figure 4.15 The state error $\tilde{x}$ caused by reducing the model (4.29) by setting $a_0 = 0$. An approximation of the state error (- -) simulated with (4.9) is compared with the actual difference (—) between the simulations of (4.29) and shown in Figure 4.10. The lower (—) plot is non-zero only for numerical reasons.

$$\frac{\partial^2 f_2}{\partial {x_1}^2} = -\frac{g}{L_2}\sin x_1 + \frac{u}{L_2}\cos x_1$$

Error bound

The gain of the system is found using (4.15) as well as (4.16), both shown in Figure 4.16. From the plot can be see that

$$\|G\| \leq 6$$

From plot Figure 4.17:

$$\| - f(\bar{x}, a) + f(\bar{x}, \bar{a})\| \leq 15$$

The gain L can be approximated in a similar way to what was done to the smaller example in (4.24). The maximum value of the control

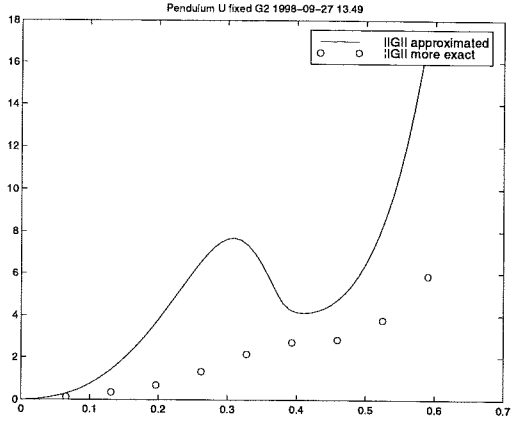

Figure 4.16 The maximum gain $\|G\|$ of the system (4.14) for the pendulum example (4.29). The gain is calculated with (4.15) (o) as well as with (4.16) (—), (4.16) gives a more conservative approximation.

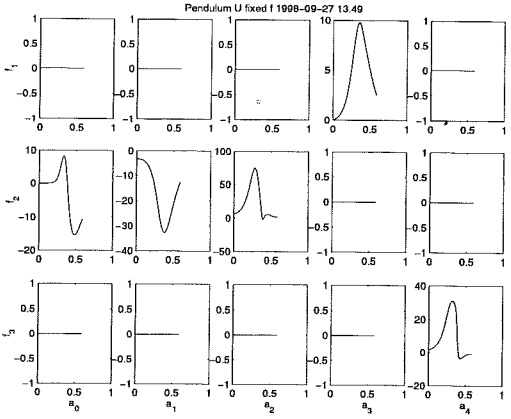

Figure 4.17 The different components of f in (4.29). The difference $f(x,a) - f(x,\bar{a})$ is shown to the left and the norm $\|f(x,a) - f(x,\bar{a})\| \leq 15$.

signal $u(t) < 32$ from Figure 2.5 is used.

$$L = \max \left\| \begin{matrix} \frac{1}{2}\tilde{x}^T \frac{\partial^2 f_1(\bar{x}+\theta_1\tilde{x},a)}{\partial x^2} \\ \vdots \\ \frac{1}{2}\tilde{x}^T \frac{\partial^2 f_n(\bar{x}+\theta_n\tilde{x},a)}{\partial x^2} \end{matrix} \right\| \leq \frac{1}{2}\tilde{x}_{max} \max \sum_{k=i}^{n} \left\| \frac{\partial^2 f_k(\bar{x}+\theta_1\tilde{x},a)}{\partial x^2} \right\| \leq$$

$$\leq \frac{1}{2}\tilde{x}_{max} \max \left\| \begin{matrix} -\frac{g}{L_2}\sin(x_1+\theta_1\tilde{x}_1) + \frac{u}{L_2}\cos(x_1+\theta_1\tilde{x}_1) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} \right\| \leq$$

$$\leq \frac{1}{2}\tilde{x}_{max}\left(\frac{g+u}{L_2}\right) \leq \frac{1}{2}\tilde{x}_{max}\left(\frac{10+32}{0.3}\right) = 70\tilde{x}_{max} \quad (4.30)$$

Figure 4.15 give that $\tilde{x}_{max}$ must be at least around 2. To use the Small Gain Theorem as in (4.12) it is necessary that $\|G\| \cdot \|L\| \leq 1$. Since $\|G\| \approx 1.2$ other approximation methods is needed to find an upper bound on the pendulum example as it is formulated here.

Ideas for future work

More sophisticated approximations of $\|L\|$ than (4.30) could be used. Knowledge about the trajectory $\bar{x}$ could be used to get a more tight approximation of $\|L\|$.

The approximation of the Hessian in (4.30) are dependent on the parameters g and L_2. If a pendulum with a much larger L_2 were studied, the approximation of $\|L\|$ would give a lower value and an error bound could be found.

An alternative would be to study a very short time interval. For a small enough time interval, $\|G\|$ would be small enough to make $\|G\| \cdot \|L\| \leq 1$. Then a strict upper bound on the error could be found.

4.6 Conclusions

In this chapter ideas are presented describing how to find an upper bound on the state error, when certain parameters are neglected in nonlinear models. Notation is introduced to capture first order as well as higher order systems. For two small examples error bound are found that are rather close to the actual error.

The pendulum example presented in Section 2.3 is studied again. The simplifications on the pendulum model are too extensive to be captured by an error bound found with the ideas from this chapter. Ideas for future work are however given, that probably would result in an upper bound for the pendulum example.

5

Conclusions

This thesis studies model reduction of nonlinear models. The models are linearized around specific trajectories. Then a simplified model is obtained based on the linearization.

Suppose there is a hypothesis that some parts of the model can be neglected. A systematic procedure is suggested to estimate the error caused by the reduction. Optimization is used to adjust remaining parameters in the model to minimize the error. The algorithm is applied to two different examples

The first example is a rotating inverted pendulum, shown in Figure 2.1. An inverted pendulum is connected to a rotating beam. The acceleration of the pivot point of the beam is considered as the control signal. The control goal is to swing up the pendulum from downward to upward equilibrium. It is shown that the pendulum, for the given swing trajectory, can be quite well approximated by a pendulum on a cart, shown in Figure 2.2. Further, it is possible to reduce the error significantly by adjusting remaining physical parameters in the pendulum model.

The second example is a drum boiler model. Both a second order and a fourth order model are studied. Input signals used for identification of a boiler at Öresundsverket in Malmö are used. For the resulting trajectory it is possible to reduce large parts of the model, adjust the remaining parameters and get a model that is almost as correct as the original one. A drum boiler is a classical control problem. It is interesting to find out that a reduced model can capture most of the behavior for a specific trajectory.

The procedures discussed above to estimate the error when reducing a system involve approximations. Therefor it is not known how good the estimation is. This thesis also presents ideas about how to find a strict upper bound for the error. The Small Gain Theorem is used on the Taylor approximation of the error. This is completed for small second order examples and the ideas are outlined for larger examples.

5.1 Future work

The results in this paper leave several questions open for further research.

Error bounds The procedure presented in Chapter 4 gives strict error bounds for small examples. It would be natural to extend this to larger examples. The derivations are done for systems in explicit form $\dot{x} = f(x,t)$ and could naturally be extended to systems in implicit form $F(x,\dot{x},t) = 0$.

Physical interpretation It would be interesting to find a physical interpretation of the parameter adjustments for the specific examples. Why can the rotating pendulum be approximated by a shorter non-rotating one. Why can the volume of steam in the drum boiler be substituted by a larger volume of water?

Fix certain parameters The current matlab code is written for examples where an adjustment coefficient a_i are put in front of every remaining parameter in a reduced model, see e.g., Equation (2.2). The formulas in this thesis however already support the possibility of putting adjustment parameters only in front of certain parameters. For physical reasons, it is often natural to fix certain parameters to one, e.g., a_3 in (2.10) and a_{11} and a_{12} in (3.10). It would be worthwhile include that possibility in the code.

Several trajectories The estimation of the error and the parameter adjustment in this paper is optimized for one trajectory. It would be a natural extension to optimize over several trajectories simultaneously.

Then it could be possible to draw conclusions about the validity of the reduced model for an operating region or a class of input trajectories.

Algebraic constrains The inverse of the Jacobian $\frac{\partial F}{\partial \dot{x}}$ is used in (2.12) to estimate the error. If the function F in (2.1) is independent of the derivatives of some states $\dot{x}_i$, then the Jacobian will not be invertible and the derivations done in this thesis would need to be modified. Such models could then be written on the form

$$0 = \begin{bmatrix} F_1(\dot{x}(t), x(t)) \\ F_2(x(t)) \end{bmatrix}$$

and a slightly modified version of the calculations would solve also this case.

Stability It would be useful to characterize how the stability properties of a system are affected when applying to the methods developed in this thesis.

6

Bibliography

ANDERSSON, L. (1999): *PhD Thesis*, chapter Introduction. Departement of Automatic Control, Lund Institute of Technology. To be published.

ÅSTRÖM, K. J. and R. D. BELL (1998): "Drum-boiler dynamics." Technical Report ISRN LUTFD2/TFRT--LUTFD2/TFRT–7577–SE--SE. Department of Automatic Control.

BELL, R. D. and K. J. ÅSTRÖM (1996): "A fourth order non-linear model for drum-boiler dynamics." In *IFAC'96, Preprints 13th World Congress of IFAC*, vol. O, pp. 31–36. San Francisco, California.

BELL, R. D. and K. J. ÅSTRÖM (1998): "Drum-boiler dynamics." *Automatica*. To be published.

BOHLIN, T. and S. F. GRAEBE (1994): "Issues in nonlinear stochastic grey-box identification." In *SYSID*. Copenhagen, Denmark.

DOYLE, J. C., B. A. FRANCIS, and A. R. TANNENBAUM (1992): *Feedback Control Theory*. Macmillan.

EUROSIM (1998): "Modeleca home page." http://www.Dynasim.se/Modelica/index.html.

JOHANSSON, R. (1993): *System modeling and Identification*. Prentice Hall.

KHALIL, H. K. (1996): *Nonlinear Systems*, 2 edition. Prentice Hall.

MAPLE (1992): *Library Reference Manual*. Springer-Verlag.

MATLAB (1996): *Matlab, The Language of Technical Computing.* The Math Works Inc.

MAVRIKIS, P. (1997): *Nonlinear Model Reduction in Time Domain.* Imperial College of Science Technology and Medicine.

MAVRIKIS, P. and R. B. VINTER (1997): "Trajectory-specific model reduction." In *36th Conference on Decision and Control.*

PERSSON, A. and L.-C. BÖIERS (1990): *Analys i en variabel.* Studentlitteratur.

PETZOLD, L. and W. ZHU (1997): "Model reduction for chemical kinetics an optimization approach." Technical Report. Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, USA.

RUDIN, W. (1976): *Principles of Mathematical Analysis.* McGraw-Hill.

RUGH, W. J. (1996): *Linear System Theory.* Prentice Hall.

SJÖBERG, J. and ET AL (1995): "Nonlinear black-box modeling in system identification: a unified overview." *Automatica*, **31:12**, pp. 1691–1722.

Part II

Paper 1

Implementation aspects of the PLC standard IEC 1131-3

Martin Öhman Stefan Johansson
Karl-Erik Årzén

Abstract IEC 1131-3 is a standard for PLCs defining four programming languages and a type of Grafcet, Sequential Function Charts (SFC). An object oriented prototype of SFC and the language Function Block Diagram has been implemented. Various execution methods are discussed. Algorithms for local and global sorting are implemented and evaluated. The standard is found to be unclear in some parts.

Keywords Grafcet, IEC 1131, Programmable logic controller, Standard.

1. Introduction

Programmable logic controllers (PLCs) are specialized computers, widely used in industrial automation. IEC 1131-3 is a standard defining four programming languages used in PLCs. This paper concerns the Function Block Diagram (FBD) and Sequential Function Charts (SFC) parts of the standard. It is based on a project Johansson and Öhman (1995) where the purpose was to make a prototype implementation of the standard and evaluate different implementation aspects. The standard is described in Section 2. The order of execution between

```
FUNCTION D : BOOLEAN

  VAR_INPUT
    A, B, C : BOOLEAN;
  END_VAR

  LD A
  OR B
  AND C
  ST D

END_FUNCTION
```

Figure 1. Instruction List

function blocks and SFC is discussed in Section 3. In Section 4 the prototype implementation is described. Different implementation aspects and alternatives are discussed. An example is shown in Section 5.

2. IEC 1131-3

This section describes IEC 1131 IEC (1995b), a PLC standard defined by the International Electrotechnical Commission (IEC). The standard contains five parts. This paper concerns part three, describing the programming languages.

IEC 1131-3 defines four different language paradigms, SFC, and program organization units. The standard also specifies their representation, and rules of evaluation.

Programming languages

The four defined language paradigms are described in this section. They are illustrated with an example, written in all four languages.

Instruction List The instruction list language is similar to assembly code with commands like LOAD and STORE. An accumulator is used to store results. This language corresponds to the programming technique that has traditionally been used in PLCs. The Instruction List language is shown in Figure 1.

```
FUNCTION D : BOOLEAN

  VAR_INPUT
    A, B, C : BOOLEAN;
  END_VAR

  D := A OR B AND C;

END_FUNCTION
```

Figure 2. Structured text

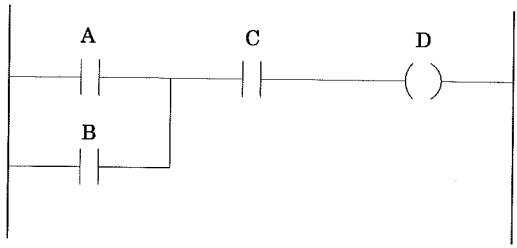

Figure 3. A ladder diagram

Structured Text The structured text language is similar to Pascal. It has sequential statements, conditional statements like IF and CASE and repetitive statements like FOR ... DO ... END_FOR and REPEAT ... UNTIL. The Structured Text language is shown in Figure 2.

Ladder Diagram The ladder diagram language specifies how to use relay ladder logic diagrams to implement boolean functions. This is a common language in modern PLCs. The Ladder Diagram language is shown in Figure 3.

Function Block Diagram In the function block diagram language, all functions, inputs and outputs are represented as graphical blocks. They are connected with lines representing the data flow. The direction is always from left to right except in feedback paths. The Function Block Diagram language is shown in Figure 4.

According to the standard it shall be possible to do jumps in all

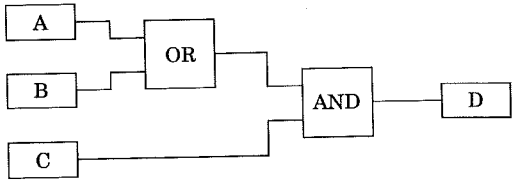

Figure 4. A function block diagram

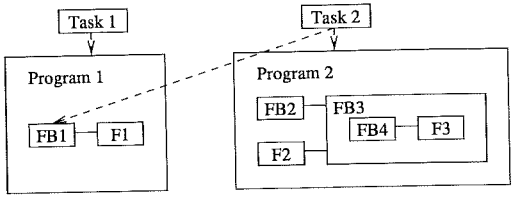

Figure 5. Program Organization Units.

languages. It is probably not so good to do jumps in the FBD language. The guidelines IEC (1995a) also advises against that.

Program Organization Units

Independent of the choice of language, PLC programming according to IEC 1131-3 uses three program organization units: functions, function blocks and programs. They are supplied by the manufacturer or defined by the user. A unit can not be recursive, i.e., it can not call itself. Figure 5 shows how the units function together.

Functions Functions have one or more inputs but only one output. A function can not store any state information, i.e., executing a function with certain values on the inputs always gives the same output value. Functions can be hierarchically defined by using already defined functions. A function can be *extensible*, meaning that the user can decide the number of inputs. The standard does not state if the user has to specify the number of inputs when the function is created or if the number can be changed during the lifetime of the function. Functions are either *typed* (operate on a specified data type) or *overloaded* (operate on different types).

Function blocks Function blocks can have several inputs and outputs. They can store state information so the values of the outputs depend on previous values.

The user must give each instance of a function block a unique name. Function blocks can be hierarchically defined by using already defined functions and function blocks.

The standard specifies that there can be multiple instances of function blocks. Probably also functions can have multiple instances.

Programs Programs are built up by functions and function blocks. Programs do not have inputs or outputs.

Tasks

The execution of programs and function blocks is controlled by tasks. The standard does not specify whether the scheduling of tasks should be preemptive or non-preemptive. A task can control more than one program organization unit and a program organization unit can indirectly be controlled by more than one task, as FB1 in Figure 5. If they try to execute at the same time and if preemptive scheduling is used, the implementation must ensure that mutual exclusion is obtained.

A task has three inputs: single, interval, and priority. It can execute once on the rising edge of *single* or periodically with the period time of *interval*. The priority of the task is set by *priority*.

Sequential Function Charts

A Sequential Function Chart (SFC) is an extended state machine, also known as a Grafcet David and Alla (1992). In the standard, SFC is a way of structuring programs and function blocks. A unit that is not structured is said to be a single action, executed continuously.

An SFC consists of two main elements, *steps* and *transitions*, shown in Figure 6. Steps can be active or inactive. The state of the SFC is determined by which steps that are active. A transition has a boolean input, *transition condition*, that can be described by any of the four languages or by a boolean variable. A transition will fire when the step above it is active and the transition condition is true. SFC in IEC 1131-3 does not support macro steps, as Grafcet normally do. An SFC can contain parallel and alternative paths, shown in Figure 6.

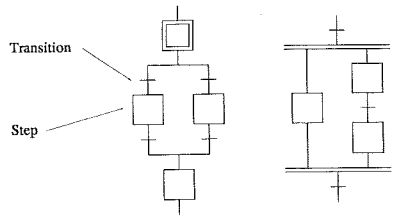

Figure 6. SFCs with alternative (left) and parallel (right) paths.

Actions It is possible to associate an *action* with a step. An action can be a single boolean variable, be described with one of the four programming languages or with an SFC. All actions must have unique names.

Action blocks Action blocks are used to associate actions with steps. An action block has a boolean input that indicates when the associated step is active. Each action block is associated with an action. Steps, actions and action blocks relate in the following way:

- A step can be connected one zero or more action blocks.
- Each action block is connected to one step.
- Each action block is associated with one action.
- Each action is associated with one or more action blocks.

Figure 7 shows that the boolean action A2 is associated with the action blocks AB1 and AB3. The non-boolean action A1 is constructed directly in the action block AB2 and associated only with that action block.

Action qualifier An action block uses an *action qualifier* to control the action. The action will execute depending on the action qualifiers on the associated action blocks and the status of the associated steps. The different action qualifiers are shown in Table 1.

Figure 7 shows action blocks, action qualifiers and actions.

N	**Non-stored** The action is active while the action block is active.
S	**Stored** The action is activated when the action block becomes active.
R	**Reset** The action is deactivated when the action block becomes active.
L	**Limited** The action is activated when the action block becomes active and is deactivated after a certain time or when the action block becomes. inactive.
D	**Delayed** The action is activated a certain time after the action block becomes active and is active as long as the action block is active
P	**Pulse** The action is active for one clock cycle when the action block is activated.
SD	**Stored and delayed** The action is activated a certain time after the action block becomes active.
DS	**Delayed and stored** The action is activated a certain time after the action block becomes active if the action block is still active.
SL	**Stored and limited** The action is activated when the action block becomes active and is deactivated after a certain time.

Table 1. Action qualifiers

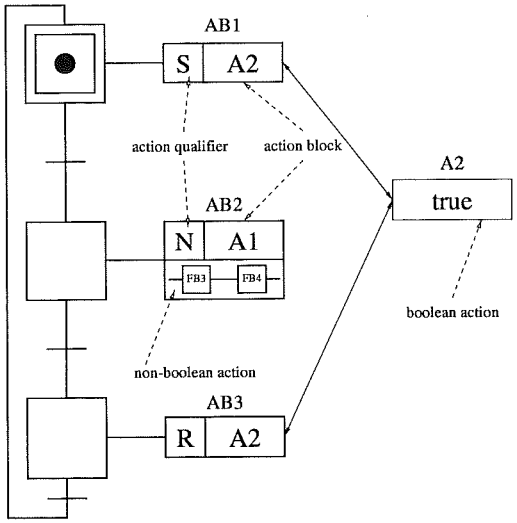

Figure 7. An SFC with action blocks, action qualifiers and actions.

3. Execution order

This section deals with the execution order between functions and function blocks, SFC-elements, action blocks, and actions. This is not well specified in the standard. It is stated that no element shall be evaluated until the states of all its inputs have been evaluated. The authors believe that this leads to the following execution order, shown in Figure 8:

1. Function blocks connected to a transition.
2. Steps and transitions.
3. Action blocks and non-boolean actions.
4. Boolean actions.

Another idea would be to execute steps and action blocks in the same phase, only executing action blocks connected to active steps.

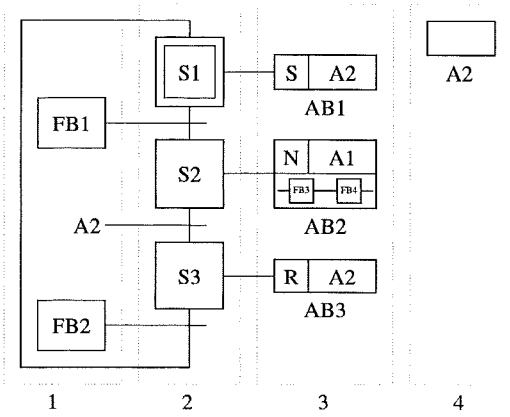

Figure 8. The execution order of functions, function blocks, steps, transitions, action blocks, and actions.

This method would however cause problems with for example the action qualifier "stored". Then the action block shall be executed also after the step has been deactivated.

Sequential Function Charts

The standard specifies that all steps should be updated synchronously. This could be achieved by going through all the steps twice. First it is checked which steps can be activated and they are marked with a flag "next", see Figure 9. Then all marked steps are activated. The order in which the steps are treated is not important. This method makes sure that each token can pass only one transition at a time.

Function blocks

Functions and function blocks are executed so that the dataflow goes from left to right. One way of storing the determined execution order is to insert the functions and function blocks in a dynamic list. The standard states that functions and function blocks can be hierarchicly defined. It is not specified in the standard if such a function or function block shall be considered as a fixed unit or if its internal structure can be changed. If a unit has a hierarchical structure, the execution

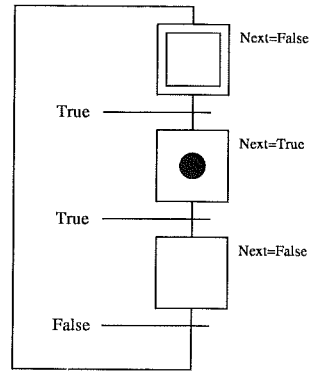

Figure 9. Execution of an SFC.

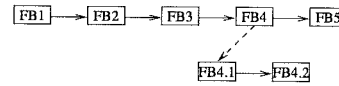

Figure 10. The function block list of the function block diagram in Figure 14 and the local list of FB4.

order can be dealt with in two ways: local and global sorting. It is not specified which one should be used.

Local sorting In local sorting each hierarchical function block have its own list containing the functions and function blocks in its internal structure. Figure 10 shows the main function block list of the FBD in Figure 14 and the local list of FB4.

The main advantage with local lists is that if changes are made in a hierarchical function block, only a new local list for that function block has to be built. This corresponds to separate compilation of program units.

Global sorting In global sorting one has a global function block list for each top level function block containing function blocks at all levels. No hierarchical function blocks are inserted in the lists, only the function blocks on their subworkspaces. Figure 11 shows the global list of the FBD in Figure 14. Note that FB4 is not in the list.

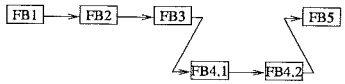

Figure 11. The global function block list of the function block diagram in Figure 14.

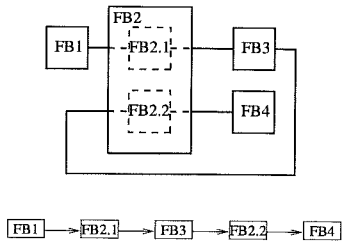

Figure 12. A loop is removed by using global lists.

The main advantage with global lists is that imaginary loops that seem to exist from a local point of view are removed. If the function block diagram in Figure 12 would be sorted locally, the user would have to insert a loop element (described in Section 4). The global list shows that this is not needed with global sorting. Removing loops gives you a faster implementation by saving clock cycles.

A disadvantage with global lists is that the implementation is more complex. When a hierarchical function block is disabled, all function blocks in its internal structure should be disabled. If they are spread out in a global list this will be difficult.

An Example PID controllers are commonly used in industrial control systems. In 1131-3 a PID controller is naturally implemented as a function block that has reference signal y_{ref} and measured variable y as inputs and the control signal v as output. However, if actuator saturations are not taken care of in the right way, the result is integrator (reset) windup, which influences the control performance negatively. A common solution to integrator windup is known as tracking. The approach is based on feeding back the saturated control signal to the PID controller and to change the integral term in the controller in such a way that the control signal generated from the controller will equal the output saturation.

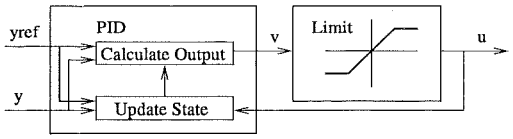

Figure 13. A PID controller.

Hence, tracking requires an additional input to the PID controller, the saturated output, often denoted u. Tracking also means that it is not possible to update the controller state until the saturated output is known, e.g. has been calculated. Sometimes the saturation limits are known in the PID controller but not always. For example, the PID controller may be a part of a cascade construct, or the actuator saturation is calculated in a special limit block, as shown in Figure 13.

The PID function block is internally decomposed into two blocks: CalculateOutput and UpdateState. The only sorting strategy that will give the correct behavior from a control point of view is global sorting. Using that strategy the execution order will be PID:CalculateOutput, Limit, PID:UpdateState. Using local sorting the old value of u will be used in the calculation of UpdateState, i.e., an unnecessary delay of one sampling interval is introduced.

General execution algorithm

Combining the discussions above gives the following general execution algorithm:

- For each function or function block (sorted locally or globally).
  - Read inputs.
  - Execute function block.
  - Write outputs.
  - Propagate output to connected function blocks.
- For each step.
  - Check if the step can be updated.
- For each step.

  - If the step can be updated.
    * Update the step.
  - Write to action block.
- For each action block.
  - Determine if the associated actions shall be active.
  - Possibly execute associated non-boolean actions.
  - Write to boolean actions.
- For each boolean action.
  - Update the action.

4. Implementation

In this section the prototype implementation is described. The implementation is done in G2, a graphical and object oriented programming environment from Gensym Corporation. G2 was chosen since it is good as a rapid prototyping tool. Different implementation methods are discussed. Most ideas can be used generally, when working in an object oriented environment. Some solutions, however, are G2 specific.

A hierarchy of classes containing tasks, function blocks, connections, steps, transitions, action blocks, actions, and global variables has been developed. The user constructs the programs on a workspace with graphical objects that are instances of these classes. Class definitions and methods are hidden for the user.

Function blocks

Since programs and functions can be seen as special cases of function blocks, only *function blocks* have been implemented. Programs are implemented as function blocks with no inputs or outputs and functions as single output function blocks without internal state information. Figure 14 shows a function block diagram.

To determine the order of execution between function blocks, a list is built when a task is initialized. The function blocks are inserted in the list in the correct execution order. Each function block is placed in the list after the function blocks connected to its input.

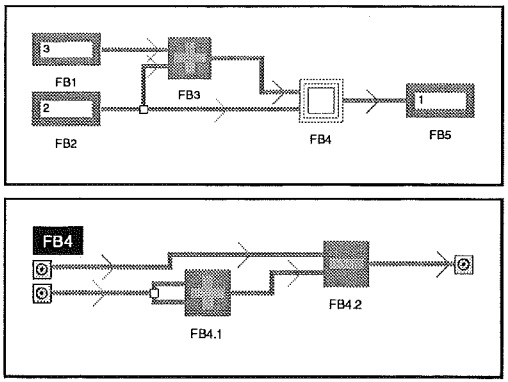

Figure 14. A hierarchical function block diagram.

Hierarchical function blocks

Hierarchical function blocks have a subworkspace where their internal structure is defined, see FB4 in Figure 14. G2 automatically makes a link from the function block to its subworkspace. It would be natural to include the internal structure in the class definition, but this is not possible in G2.

Hierarchical function blocks are inserted in execution lists using either local or global sorting.

Algebraic loops

The order in which the function blocks are executed must follow specific rules. A function block can not be executed until the function blocks connected to all its inputs have been executed. This rule causes problems when building a loop. In Figure 15 FB1 must be executed before FB2 to give the input of FB2 a value. For the same reason FB2 must be executed before FB1. This problem is called an *algebraic loop*.

The problem is solved by introducing a so called *loop element*. The user decides where to break the loop and insert a loop element. A loop element has the special property that it can be executed before function blocks connected to its input. It reads the old value from the input and writes it to the output. Therefore, the user can use them, instead of variables, as a memory. In Figure 16 the algebraic loop problem is

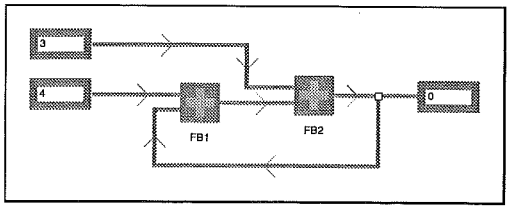

Figure 15. Algebraic loop.

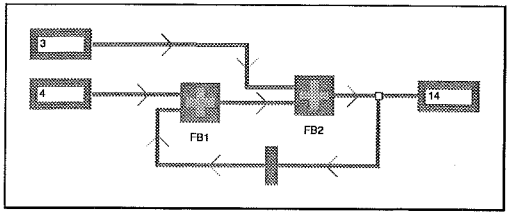

Figure 16. Loop element.

solved by inserting a loop element and FB1 can now be executed before FB2.

Sequential Function Charts

SFC elements include steps, transitions and branches. Branches are used to build alternative and parallel paths. All the elements have methods, used to determine whether a step should be activated or deactivated.

All steps and transitions have an input and an output, used to connect them with each other. Steps are represented graphically as rectangles. A filled red circle indicates when a step is active. The state of the step is written to a boolean output that can be connected to an action block.

Transitions are represented graphically as a horizontal bar. They have a boolean input that acts as a transition condition. Figure 17 shows an SFC.

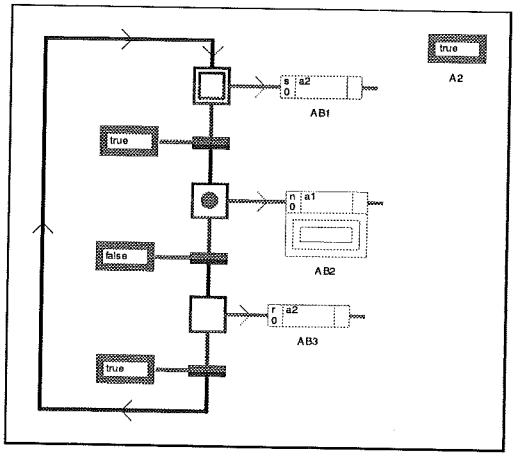

Figure 17. An SFC with steps, transitions, action blocks and actions.

Global variables and actions

Global variables of different data types are implemented. Actions are a subclass of boolean variables. To get the value of a variable, special *get-var* functions are used. If the action only function as a boolean variable it has a graphical representation and is placed on an arbitrary workspace by the user. If the action is defined with function blocks or SFC, however, it is constructed on a subworkspace of that action block and can then be associated only with that action block. The standard defines when an action should be active depending on the action qualifiers of the associated active action blocks. This functionality is implemented in a method of the class action.

When an action is defined on the subworkspace of an action block, the action block functions like a hierarchical function block with an enable signal associated with the action. The function blocks on the subworkspace should only be executed when the action block is enabled. This is solved by placing those function blocks in local lists even when the rest of the resource is sorted globally, i.e., total global sorting is not implemented.

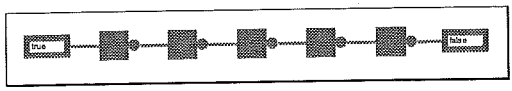

Figure 18. A test example.

Figure 19. The function block list list of the test example in Figure 18.

Tasks

To control the tasks a *task handler* is defined. Each time unit the task handler decreases a counter in each task. When the counter reaches zero the task will execute and reset the counter to the execution interval.

When a task executes it runs all its associated function blocks. Hence, a link from the task to the associated function blocks is needed. A natural solution would be to let the task have a static list of pointers to all its function blocks, but G2 does not support pointers. Instead, each function block stores the name of its task and when a task is activated G2 effectively inserts all associated function blocks in a list.

Execution procedure

Execution of elements could also be handled by replacing the lists with a procedure that would be generated automatically when the task is activated. The list would contain the code from all the function blocks, inserted in the correct order, using either local or global sorting. Each time a task is executed it would only execute the procedure. A small example was made to evaluate the time difference, see Figure 18.

The execution time between a function block list, Figure 19, and a procedure, Figure 20 when executed 100 000 times were compared. Execution with the list took 112 s and with the procedure 7 s. This shows that there is a lot to gain using procedures.

Asynchronous dataflow

Another solution would be to let the execution be asynchronously controlled by the data flow. Each function block would then be executed

```
begin
   x = true;
   x = not (x);
   x = not (x);
   x = not (x);
   x = not (x);
   x = not (x);
end;
```

Figure 20. The procedure of the test example in Figure 18.

as soon as the value of any of its inputs is changed. This does not correspond well to the standard since it is stated that each function block should be executed every task cycle, independently of if its input values have changed or not.

5. An Example

This section describes how FBD and SFC could be combined in a larger and slightly more realistic example. The process is simulated in G2. Figure 21 shows the process that contains a tank, a pump, a valve and a heater.

The example contains two PID-controllers, one controling the level of the tank and one controling the temperature. To increase the level the pump is used and to increase the temperature the heater is used. The valve is used to decrease the level. If the valve is open and the level is kept constant, new cold water must be added. This means that the valve is used indirectly to decrease the temperature. The reference value of each controller is set by an action. The action *fill* sets the reference level to 15, while the default value is 5. The action *heat* sets the reference temperature to 30, while the default value is 20.

The main process loop is represented as an SFC, shown in Figure 22. In the initial step *fill* is activated and starts to fill the tank. When the tank is filled to the level 10 the step will be deactivated and the two steps in the parallel paths will be activated. The left step keeps the action *fill* active so that the level is kept around 15. The step

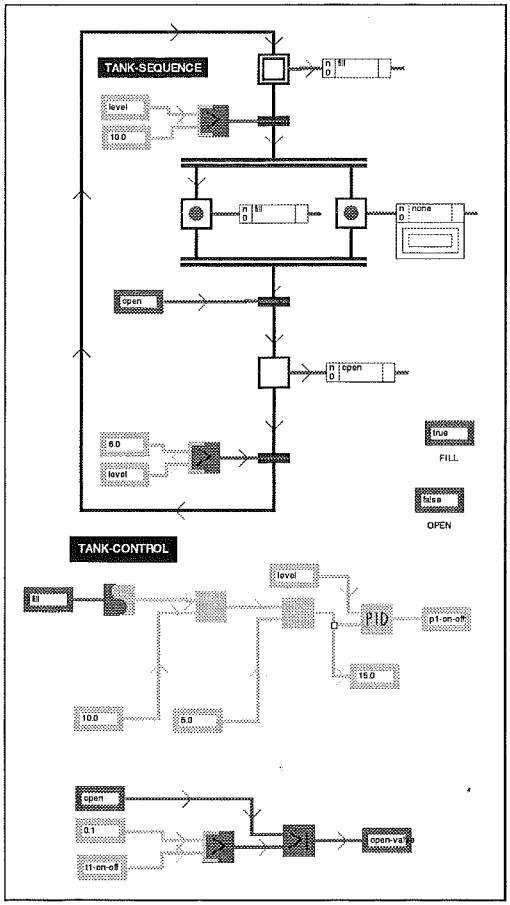

Figure 22. The main process control

6. Conclusions

The standard IEC 1131-3 contains most of the needed information but is sometimes not easy to understand. It is well specified how SFC and FBD function separately, but not so well how they function together.

The execution order between function blocks and SFC elements is the most unclear part the standard. Probably, the parts of a FBD connected to transitions should be executed before an SFC and parts associated with actions after the SFC.

In the prototype implementation, all function blocks are sorted into lists before they are executed. Algorithms for sorting the lists locally or globally are implemented. Local sorting was found to be easier to implement. However, global sort is necessary if artificial time delays are to be avoided.

For a small example, a single procedure is written with the code from all the function blocks in a function block diagram. The program executed an order of magnitude faster with this method. It would be interesting to implement methods for generating such procedures.

The programming environment G2 is useful as a prototype tool since it is easy to learn and work with. Since it is slow and quite expensive, it is however unrealistic to use it for real PLCs.

7. Bibliography

DAVID, R. and H. ALLA (1992): *Petri Nets and Grafcet: Tools for modelling discrete events systems.* Prentice-Hall.

ÖHMAN, M., S. JOHANSSON, and K. E. ÅRZÉN (1997): "Implementation aspects of the PLC standard iec 1131-3." In *CACSD'97.* IFAC.

IEC (1995a): "Guidelines for the application and implementation of programming languages for programmable controllers." Technical Report. IEC.

IEC (1995b): "IEC 1131-3." Technical Report. IEC. first edition.

JOHANSSON, S. and M. ÖHMAN (1995): "Prototype implementation of the PLC standard IEC 1131-3." Master thesis ISRN LUTFD2/TFRT--5547--SE. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

LEWIS, R. W. (1995): *Programming industrial control systems using IEC 1131-3.* The Institution of Electrical Engineers.

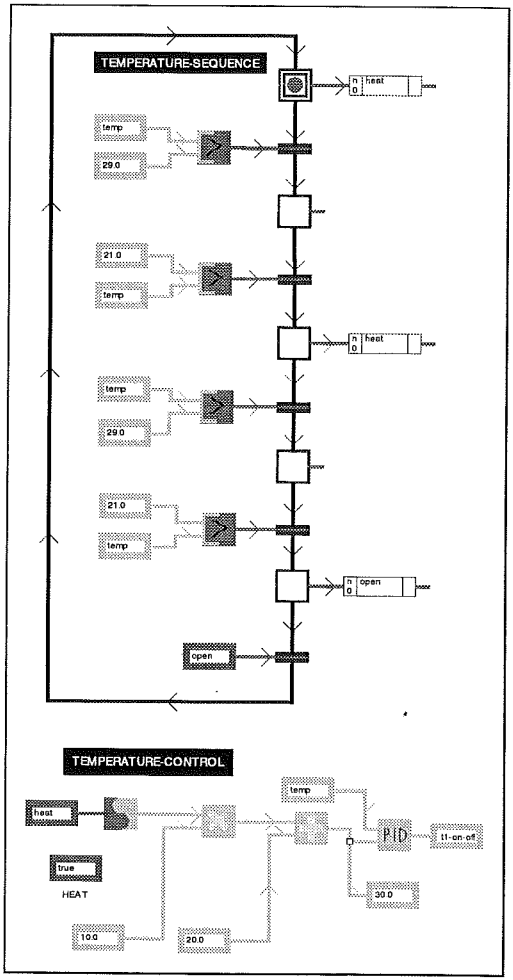

Figure 23. The control of the temperature sequence