Linda Brus

Nonlinear Identification and Control with Solar Energy Applications
Abstract

Nonlinear systems occur in industrial processes, economical systems, biotechnology and in many other areas. The thesis treats methods for system identification and control of such nonlinear systems, and applies the proposed methods to a solar heating/cooling plant.

Two applications, an anaerobic digestion process and a domestic solar heating system are first used to illustrate properties of an existing nonlinear recursive prediction error identification algorithm. In both cases, the accuracy of the obtained nonlinear black-box models are comparable to the results of application specific grey-box models. Next a convergence analysis is performed, where conditions for convergence are formulated. The results, together with the examples, indicate the need of a method for providing initial parameters for the nonlinear prediction error algorithm. Such a method is then suggested and shown to increase the usefulness of the prediction error algorithm, significantly decreasing the risk for convergence to suboptimal minimum points.

Next, the thesis treats model based control of systems with input signal dependent time delays. The approach taken is to develop a controller for systems with constant time delays, and embed it by input signal dependent resampling; the resampling acting as an interface between the system and the controller.

Finally a solar collector field for combined cooling and heating of office buildings is used to illustrate the system identification and control strategies discussed earlier in the thesis, the control objective being to control the solar collector output temperature. The system has nonlinear dynamic behavior and large flow dependent time delays. The simulated evaluation using measured disturbances confirm that the controller works as intended. A significant reduction of the impact of variations in solar radiation on the collector outlet temperature is achieved, though the limited control range of the system itself prevents full exploitation of the proposed feedforward control. The methods and results contribute to a better utilization of solar power.

Keywords: feedforward control, model predictive control, nonlinear control, nonlinear systems, recursive identification, solar power, system identification, time delay systems

© Linda Brus 2008

ISSN 1104-2516
urn:nbn:se:uu:diva-8594 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-8594)
Acknowledgments

First of all I would like to express my gratitude to my advisors Prof. Bengt Carlsson and Prof. Torbjörn Wigren for taking me on as a PhD student, for sharing their profound knowledge, and providing me with guidance and encouragement during this process. I would also like to thank my other coauthor Dr Darine Zambrano for fruitful collaboration and interesting discussions.

A special thanks goes to Prof. Graham C. Goodwin for taking the time to read my work and come half way around the world to be my faculty opponent.

Further, I would like to express my gratitude to organizations that have supported me financially: Uppsala University Ångpanneföreningens forskningsstiftelse, and Bernt Jämarks stiftelse för vetenskaplig forskning, by giving me grants, SIDA for support through the Swedish-South African research partnership programme (Control-Aided Parameter Estimation in Oscillatory Systems), and the EC 6th Framework programme for support through Specific Targeted Research or Innovation Project (HipCon, Contract no. NMP2-CT-2003-505467, and EU HYCON Network of Excellence, Contract no. FP6-IST-511368).

To my colleagues at SysCon and TDB: I will consider myself lucky if I ever find workmates with whom I can have lunch-discussions that are half as twisted as ours have been. Some of you also deserve a special thanks for helping me keep my sanity. You know who you are.

Finally I would like to thank my family and friends, for seemingly endless love, support, and - when I need it most - perspective!
Contents

Acknowledgments 3

Olinjär identifiering och styrning med solenergitillämpningar 9

Glossary 13

1 Introduction 15
  1.1 Thesis Outline ........................................... 16

I Background 21

2 Technical Starting Point 23
  2.1 Nonlinear System Identification .............................. 23
    2.1.1 Grey-box Identification .................................. 24
    2.1.2 Black-box Identification .................................. 24
    2.1.3 Modeling Approaches ..................................... 27
    2.1.4 Batch and Recursive Identification ..................... 30
    2.1.5 Analysis .................................................. 31
  2.2 Nonlinear Control .......................................... 32
    2.2.1 Reusing Linear Control ................................... 32
    2.2.2 Nonlinear Feedback Control ............................... 34
    2.2.3 Nonlinear Feedforward Control ........................... 37
  2.3 Time Delays ................................................ 39
    2.3.1 Time Delays in System Identification .................... 39
    2.3.2 Time Delays in Control .................................. 40
  2.4 Solar Energy for Heating and Cooling ...................... 42
    2.4.1 Cooling by Absorption ................................... 43
    2.4.2 Solar Collector Dynamics .................................. 43
    2.4.3 Control of the Solar Collector Outlet Temperature ..... 44
3 An RPEM for Identification of Nonlinear Systems 47
  3.1 Model Structure ........................................... 47
  3.2 Discretization ........................................... 49
  3.3 Algorithm ........................................... 50
  3.4 Scaling of the Sampling Period .......................... 51

II Contributions 53

4 Using the Nonlinear RPEM 55
  4.1 Anaerobic Digestion ........................................ 55
    4.1.1 System Description .................................... 56
    4.1.2 Experiments ........................................... 56
    4.1.3 Discussion ........................................... 60
  4.2 Solar Heating ........................................... 63
    4.2.1 System Description .................................... 63
    4.2.2 Experiments ........................................... 65
    4.2.3 Discussion ........................................... 71
  4.3 Summary .................................................. 72

5 Convergence Analysis 73
  5.1 Model Structure ........................................... 73
  5.2 Algorithm Modifications ................................... 74
  5.3 Associated ODE Analysis Tools ............................ 76
  5.4 Conditions on the Algorithm and the Data .................. 77
  5.5 Results .................................................. 79
    5.5.1 Outline of Proof ..................................... 79
    5.5.2 Establishing the Tools for Analysis ................. 79
    5.5.3 The Main Result ..................................... 81
  5.6 Numerical Example ....................................... 83
  5.7 Summary .................................................. 84

6 An Initialization Algorithm 87
  6.1 A Kalman Filter Based Identification Algorithm ............ 88
    6.1.1 Generation of Regressors .............................. 88
    6.1.2 Algorithm ........................................... 89
  6.2 Experiments ............................................. 90
    6.2.1 Accuracy Comparison .................................. 90
    6.2.2 Application of the Algorithm for Initialization .... 96
    6.2.3 Results with Live Data ................................ 98
  6.3 Initialization Software ................................... 103
    6.3.1 RPEM Software Package Overview ...................... 103
    6.3.2 Initialization Software Overview ..................... 103
  6.4 Summary .................................................. 106
## 7 MPC for Systems with Long Input Signal Dependent Delays 107

7.1 Optimal Control of Systems with Constant Time Delay . . . . 107
7.1.1 Two Optimal Control Problems . . . . . . . . . . . . . . 108
7.1.2 Equivalence of the Two Control Problems . . . . . . . . 111
7.1.3 Controller Design Algorithm . . . . . . . . . . . . . . . . . 112
7.2 Input Dependent Sampling . . . . . . . . . . . . . . . . . . . 113
7.2.1 Algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 113
7.3 MPC . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 113
7.3.1 The MPC Principle . . . . . . . . . . . . . . . . . . . . . . . 113
7.4 Optimal Feedforward MPC Software . . . . . . . . . . . . . . 114
7.4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
7.4.2 Optimization Horizon . . . . . . . . . . . . . . . . . . . . . . 116
7.4.3 Software Package Overview . . . . . . . . . . . . . . . . . . 117
7.4.4 Code Validation . . . . . . . . . . . . . . . . . . . . . . . . . . . 117
7.5 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 120

## 8 Identification and Control of a Solar Cooling Plant 125

8.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 125
8.2 The Solar Powered Air–conditioning Plant . . . . . . . . . . . . 126
8.2.1 Plant Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126
8.2.2 Solar Collector Dynamics . . . . . . . . . . . . . . . . . . . . . 128
8.2.3 Control Objective . . . . . . . . . . . . . . . . . . . . . . . . . . . 129
8.3 Nonlinear System Identification . . . . . . . . . . . . . . . . . . 129
8.3.1 Identification Method Selection . . . . . . . . . . . . . . . . 129
8.3.2 Requirements on Measurements . . . . . . . . . . . . . . . . 130
8.3.3 Off-line Time Variable Delay Compensation of the Data 131
8.4 Identification Results . . . . . . . . . . . . . . . . . . . . . . . . . 135
8.4.1 Solar Collector Dynamics . . . . . . . . . . . . . . . . . . . . . 135
8.5 Nonlinear Feedforward MPC of the Solar Cooling Plant . . . 141
8.5.1 Controller Overview . . . . . . . . . . . . . . . . . . . . . . . . . 142
8.5.2 Flow Dependent Sampling . . . . . . . . . . . . . . . . . . . . 142
8.5.3 Extended Integrating Plant Model . . . . . . . . . . . . . . . 143
8.6 Feedforward Control Results . . . . . . . . . . . . . . . . . . . . 145
8.7 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 148

## 9 Conclusions 151

9.1 Contributions and Results . . . . . . . . . . . . . . . . . . . . . . 151
9.2 Future Research . . . . . . . . . . . . . . . . . . . . . . . . . . . . 153

### III Appendices 155

**A Proof of Theorem 1 (Chapter 5)** 157

A.1 Proof of Lemma 1 - Reformulation of (5.4) . . . . . . . . . . . . 157
A.2 Verification of Regularity Conditions . . . . . . . . . . . . . . . 158
A.3 Verification of the Boundedness Condition . . . . . . . . . . . . 161
Olinjär identifiering och styrning med solenergi-tillämpningar

Här följer en kort sammanfattning, på svenska, av avhandlingens innehåll. Avhandlingen sorteras under ämnet elektroteknik med inriktning reglerteknik. Först ges en kortfattad introduktion till de ämnen som behandlas i avhandlingen, för att ge en bakgrund till vad avhandlingen handlar om och för att motivera varför dessa ämnen studeras. Sedan följer en mer specifik redogörelse av innehållet i avhandlingen.

Vad handlar avhandlingen om?

Olinjära dynamiska system förekommer i industriella processer, ekonomiska system, bioteknik och inom många andra områden. Ofta finns mycket att vinna för den som kan skapa sig en bild av den underliggande dynamiken, och om möjligt styra systemet i fråga. Många har t ex försökt förstå och förutspå fluktuationerna i aktiekurser med hopp om att kunna ligga steget före resten av marknadens aktörer. Möjligheten att kunna skapa en bra modell av ett förlopp kan också utgöra ett sätt att anpassa avancerad medicinsk behandling till specifika individer. På så sätt kan man i viss mån minska behovet av att prova sig fram vid t ex dosering av läkemedel, utan kan basera dosen på individuella egenskaper. Generellt kan modellering och system identifiering användas för att förstå komplicerade samband som inte förefaller uppenbara vid första påseende.

Många industrier har hundratals eller t o m tusentals regulatorer för att styra processer och se till att den slutliga produkten håller en jämn kvalitet och att de tillgängliga resurserna utnyttjas på ett effektivt sätt. Exempel på storheter som styrs på detta sätt är t ex tjockleken på papper i pappersframställning och farthållare i olika typer av fordon, men även utsläpp av föroreningar från industrier och näringsämnen från reningsverk måste styras med liknande metoder. Ett annat vanligt exempel på ett enkelt reglersystem är radiatorer kopplade till en termostat där temperaturen i lokalen ska hållas
på ett konstant värde. För andra tillämpningar, som till exempel solvärme- och solkylsystem, hjälper modeller och regulatorer till att maximera utvinnningen av den ren energikällan, och därigenom minska behovet av andra, mindre ren, energislag.


Översikt över avhandlingen

Avhandlingen inleds med en teknisk bakgrund till de ämnen som behandlas senare; olinjär systemidentifiering, styrning av olinjära system, hantering av tidsfördröjningar, samt ett avsnitt om solenergi som beskriver hur systemidentifiering och reglerteknik använts för denna tillämpning. Först illustreras användningen av olinjär identifiering med hjälp av en tidigare algoritm, ap- plicerad på två exempel; en anaerob nedbrytningsprocess (biogasproduktion) och det småskaliga solvärmesystem som nämns ovan. I biogasexemplet blir den slutliga modellen av systemet avsevärt mindre kompleks än den modell baserad på fysikaliska samband som använts för att generera data till experimenten. I solvärmesystemet, där mängden data är relativt liten, används multipla svep av algoritmen över data för att få fram en godtagbar modell.

För många olinjära metoder är konvergens en faktor som kan komplicerar användandet av en algorit. Därför genomförs en matematisk konvergensanalyser som resulterar i formulering av ett antal villkor på algoritm och data som är tillräckliga för att uppnå konvergens.

Användandet av algoritmen i de två exemplen, tillsammans med konvergensanalysen indikerar ett behov av en metod för att hitta bra startvärd för algoritmens parametersökning. En sådan metod föreslås och utvärderas i avhandlingen.

En modellbaserad regulator för system med insignalberoende tidsfördröjningar utvecklas sedan. Först visas att två optimeringsproblem, där det ena kräver
avsevärt mycket mindre beräkningar än det andra, ger identiska lösningar. Vidare diskuteras hur den styrsignalberoende tidsfördröjningen kan hanteras med omsampling, och slutligen implementeras regulatorn som ett mjukvarupaket för specialfallet där systemets tidsfördröjning är omvänt proportionell mot styrsignalen. Det minst komplexa optimeringsproblemet används därvid.

Som en slutlig illustration används de metoder för identifiering och styrning som diskuterats i avhandlingen på det kombinerade solvärme- och solkylesystemet i Sevilla. Syftet med experimenten är att skapa en modell av den oljära dynamiken i solfångarna och utveckla en regulator som motverkar variationer i vattentemperaturen ut från solfångarna. Identifieringen och regulatordesignen kompliceras av långa varierande tidsfördröjningar som påverkas av flödet genom solfångarna, vilket också utgör systemets styrsignal. Resultaten är mycket lovande, det visar sig möjligt att kompensera bort mycket av temperaturvariationerna som uppkommer i solfångarna under perioder med växlande molnighet. Denna situation utgör sannolikt det mest besvärliga driftförhållandet.
Glossary

The following list of notation and abbreviations are intended to introduce symbols and acronyms that are frequently used throughout this thesis.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M^T, v^T$</td>
<td>Transpose of matrix $M$ and vector $v$ respectively</td>
</tr>
<tr>
<td>$T_s$</td>
<td>Sampling period</td>
</tr>
<tr>
<td>$q$</td>
<td>Forward shift operator</td>
</tr>
<tr>
<td>$q^{-1}$</td>
<td>Backward shift operator ($q^{-1}s(t) = s(t - T_s)$)</td>
</tr>
<tr>
<td>$\dot{x}(t)$</td>
<td>Time derivative of $x(t)$</td>
</tr>
<tr>
<td>$u_{i}^{(k)}(t)$</td>
<td>$k^{th}$ time derivative of $u_{i}(t)$</td>
</tr>
<tr>
<td>$\hat{\theta}(t)$</td>
<td>Scaled parameter estimate</td>
</tr>
<tr>
<td>$\hat{\theta}_{rescaled}(t)$</td>
<td>Rescaled parameter estimate</td>
</tr>
<tr>
<td>$\nabla f$</td>
<td>Gradient of $f$</td>
</tr>
<tr>
<td>$\mathcal{L}$</td>
<td>Laplace operator</td>
</tr>
<tr>
<td>$\in$</td>
<td>Belongs to the set</td>
</tr>
<tr>
<td>$\subseteq$</td>
<td>Subset of</td>
</tr>
<tr>
<td>$\triangleq$</td>
<td>Equal by definition</td>
</tr>
<tr>
<td>$\mathbb{R}^n$</td>
<td>The real $n$-dimension space</td>
</tr>
<tr>
<td>$\lambda_i(M)$</td>
<td>The $i^{th}$ eigenvalue of $M$</td>
</tr>
<tr>
<td>$E$</td>
<td>Expectation operator</td>
</tr>
<tr>
<td>$I$</td>
<td>Identity matrix of unspecified dimensions</td>
</tr>
<tr>
<td>$I_n$</td>
<td>Identity matrix of dimension $(n \times n)$</td>
</tr>
<tr>
<td>$\partial D_M$</td>
<td>The boundary of the set $D_M$</td>
</tr>
<tr>
<td>$D_M \setminus \partial D_M$</td>
<td>The interior of $D_M$</td>
</tr>
<tr>
<td>$f_x, L_x$</td>
<td>Derivative w.r.t. $x$ ($\frac{df}{dx}, \frac{dL}{dx}$)</td>
</tr>
<tr>
<td>$\text{col}(M)$</td>
<td>Generates a vector by stacking the columns of $M$</td>
</tr>
<tr>
<td>$\text{diag}(v)$</td>
<td>Diagonal matrix where the diagonal elements are given by $v$</td>
</tr>
<tr>
<td>$\text{vec}(v), \text{vec}(M)$</td>
<td>Generates a vector/matrix by stacking the rows in $i$</td>
</tr>
</tbody>
</table>
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/D</td>
<td>Analog-to-Digital</td>
</tr>
<tr>
<td>ADM1</td>
<td>Anaerobic digestion model no. 1</td>
</tr>
<tr>
<td>ARMAX</td>
<td>Autoregressive moving average with exogenous input</td>
</tr>
<tr>
<td>ARX</td>
<td>Autoregressive with exogenous input</td>
</tr>
<tr>
<td>CRLB</td>
<td>Cramér Rao lower bound</td>
</tr>
<tr>
<td>D/A</td>
<td>Digital-to-Analog</td>
</tr>
<tr>
<td>dB</td>
<td>decibel (10\log_{10}(\cdot))</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite impulse response</td>
</tr>
<tr>
<td>HJB</td>
<td>Hamilton-Jacobi-Bellman</td>
</tr>
<tr>
<td>i.i.d.</td>
<td>independent and identically distributed</td>
</tr>
<tr>
<td>LQG</td>
<td>Linear quadratic Gaussian control</td>
</tr>
<tr>
<td>LS</td>
<td>Least squares</td>
</tr>
<tr>
<td>MIMO</td>
<td>Multiple input multiple output</td>
</tr>
<tr>
<td>MISO</td>
<td>Multiple input single output</td>
</tr>
<tr>
<td>MPC</td>
<td>Model predictive control</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean square error</td>
</tr>
<tr>
<td>NARMAX</td>
<td>Nonlinear autoregressive moving average with exogenous input</td>
</tr>
<tr>
<td>NARX</td>
<td>Nonlinear autoregressive with exogenous input</td>
</tr>
<tr>
<td>NFIR</td>
<td>Nonlinear finite impulse response</td>
</tr>
<tr>
<td>NOE</td>
<td>Nonlinear output error</td>
</tr>
<tr>
<td>ODE</td>
<td>Ordinary differential equation</td>
</tr>
<tr>
<td>OE</td>
<td>Output error</td>
</tr>
<tr>
<td>PDE</td>
<td>Partial differential equation</td>
</tr>
<tr>
<td>PEM</td>
<td>Prediction error method</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional integral derivative (controller)</td>
</tr>
<tr>
<td>PRBS</td>
<td>Pseudo random binary sequence</td>
</tr>
<tr>
<td>RHS</td>
<td>Right hand-side</td>
</tr>
<tr>
<td>RPEM</td>
<td>Recursive prediction error method</td>
</tr>
<tr>
<td>SISO</td>
<td>Single input single output</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>w.p.1</td>
<td>with probability 1</td>
</tr>
<tr>
<td>w.r.t.</td>
<td>with respect to</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Nonlinear dynamic systems are abundant in numerous industrial processes, economical systems, biotechnology, as well as in many other applications. Needless to say there is often a lot to gain for those who can understand the underlying dynamics through modeling and when possible, control the system of interest. There are, for example, lots of people who spend a considerable amount of time trying to model the fluctuations of the stock market, with the hopes of gaining advantages compared to others. Many industries have hundreds or thousands of controllers to ensure that the final product has an even quality and that no excess resources are wasted. For other applications, like solar heating and cooling systems, models and controllers help maximize the utilization of the clean energy source, thereby reducing the need for other, possibly more polluting, energy sources.

This thesis treats nonlinear system identification as well as nonlinear control based on identified models. The starting point is a recursive prediction error algorithm (RPEM) for identification of nonlinear state space models described in [97]. The algorithm utilizes a nonlinear ordinary differential equation (ODE) model structure that can be used to describe a large variety of nonlinear systems.

In the thesis the use of the algorithm from [97] is first illustrated through identification experiments using data from two nonlinear systems; an anaerobic digestion process where the final model structure is much less complex than the first principles based model from which the data was generated, and a domestic solar heating system where the data set is relatively small for use with a recursive method.

As for many nonlinear system identification methods, convergence is a factor that may complicate the use of a chosen algorithm. Therefore the thesis presents a convergence analysis based on averaging techniques that leads up to the formulation of sufficient conditions for convergence of the algorithm of [97].

The experience of running the algorithm in the above examples and the convergence analysis both point to a common problem in nonlinear identifica-
tion; how to avoid convergence to suboptimal minima of a non-convex criterion function. A method for finding initial parameters for the RPEM is therefore proposed and the properties of the initialization algorithm are evaluated.

A model based optimal feedforward controller for systems with flow variant time delays is then developed. Contrary to many other contributions, the thesis uses a nonlinear dynamic model. Two different choices of criterion functions are proven to give equivalent solutions, but one of the criteria imply a significantly lower computational load when searching for the optimal control sequence. The controller is implemented in MATLAB for a special case where the system has a time delay that depends on the inverse of the input. The delay variation is handled through an input dependent resampling of the data, which makes the delay constant as measured in the number of samples inside the controller, and consequently straight forward to handle.

As a final contribution the methods for identification and control discussed earlier in the thesis are applied to a solar cooling system in Spain. The aim of the experiments is to model the nonlinear dynamic behavior of a solar collector and develop a feedforward controller for it. The identification and control design is complicated by large and varying time delays that occur as the transportation time through pipes and solar collectors are affected by the flow of the heat transportation medium, which also acts as a control signal of the system. In particular it is found that among a large number of nonlinear models, a linear model with nonlinear disturbance terms performs the best. This contributes to motivate a combination of nonlinear feedforward and linear feedback control.

1.1 Thesis Outline

The contents of this thesis is organized as follows. Chapter 2 and 3 provide a technical background to the topics treated later in the thesis. These two chapters consist solely of work done by others, and are included to give the reader a sense of the context of the contributions of this thesis. Chapter 4-8 contain contributions based on research completely or partly performed by the thesis author.

Chapter 2: Technical Starting Point

This chapter gives a brief introduction to nonlinear system identification, nonlinear control, the effect of time delays on a system, and the utilization of solar energy for heating and cooling purposes. The chapter serves as a starting point for the problems studied later in the thesis.

Chapter 3: An RPEM for Identification of Nonlinear Systems

The nonlinear identification work performed in the thesis is to a large extent based upon or motivated by properties of an algorithm proposed by [97]. In this chapter the algorithm and the model structure upon which it is based are presented.
Chapter 4: Using the Nonlinear RPEM

The use of the RPEM algorithm from the previous chapter is explained and exemplified using two systems; a solar heating system and an anaerobic digestion process. Chapter 4 is based on [15; 16]:


Chapter 5: Convergence Analysis

An analysis of the convergence properties of the RPEM from Chapter 3 is performed using averaging techniques, and conditions that imply convergence of the algorithm are formulated. The analysis serves as a tool to provide the user with a sense of when the algorithm can be expected to work as intended. Chapter 5 is based on [17]:


Chapter 6: An Initialization Algorithm

Based on the results of the convergence analysis in Chapter 5 it can be argued that a method for finding initial parameters for the RPEM algorithm is required to reduce the risk of convergence to suboptimal minima of the criterion function. Such a method is proposed and the properties of the suggested method are discussed. Chapter 6 is based on [19; 21; 99]:


Chapter 7: MPC for Systems with Long Input Signal Dependent Delays

A discussion on how the models obtained with the methods in Chapter 3-6 can be used for model predictive control is presented. Much of the treatment is valid for feedback and feedforward control. Results on the impact of the choice of optimal control criterion from [98] are generalized to the MIMO case. Further, the handling of varying time delays using input dependent sampling is discussed. Chapter 7 is partly based on [18; 20; 22; 98]


Chapter 8: Identification and Control of a Solar Cooling Plant

The chapter presents the main application of the thesis; identification and control of solar collectors used for cooling at the University of Seville, Spain. The solar collectors have nonlinear dynamics. Furthermore, the time delay from input and measurable disturbances to the output are large and input signal dependent. Chapter 8 is based on [18; 21; 23]:


Chapter 9: Conclusions

The chapter presents the main contributions and results of the thesis. Thereafter, topics for future research, related to the subjects treated in the thesis are discussed.
1. Introduction
Part I

Background
Technical Starting Point

2.1 Nonlinear System Identification

System identification concerns mathematical modeling of dynamic systems based on measured data. The use of measured data makes the method inherently experimental, and the objective is normally to obtain a model that describes the behavior of the original system sufficiently well for the model to serve its purpose. Such purposes can be anything from an increased understanding of the underlying dynamics of the system to simulations, tracking of dynamics, fault detection, and controller design. The generality of system identification makes it applicable in industrial processes and biotechnology, as well as chemistry and economy, just to mention a few examples.

Many systems have nonlinear dynamics, which complicates the modeling. Some examples of applications where nonlinearities occur include pH control [71], control valves [91], flight dynamics [36], and power systems [5]. In certain cases a linear model may be sufficient to describe the system, at least around some operating point. However, it is e.g. shown in [35] that linear models may be sensitive even to small nonlinearities, in which cases standard validation tools may not give a correct image of how well the the model actually describes the system. In other cases, e.g. flight dynamics [36], the behavior of the system varies so much over the allowed operating range that gain-scheduling [3] or adaptive control schemes [85] are required. For such systems it may be advantageous to use nonlinear models with a wider operating range.

Another motivation for the study of nonlinear systems is the use of model based nonlinear control. A large variety of systematic design methods based on nonlinear ordinary differential equation (ODE) models have emerged in the last two or three decades. Feedback linearization and backstepping [44; 65] are two examples of the more important methods. To design controllers like these a nonlinear ordinary differential equation model of the system is usually required. Consequently, tools from the system identification field for producing such models become highly interesting. The methods for identifying nonlinear
ODE models that are discussed in this thesis fit directly into the framework of the control methods described in e.g. [44; 65].

Both linear and nonlinear identification methods can be described as being either of black-box or grey-box type. In a grey-box, or semi-physical method, the modeling is performed using a priori knowledge of the physical properties of the system. Unlike grey-box models, a black-box, or non-physical, model is a mathematical description of a system, where little or no consideration is taken to the physical connection between different system variables. This implies that there may not be a complete physical interpretation of each part of the black-box model, which may under certain circumstances be considered a drawback. Clearly, little a priori knowledge of the system is required, and since the model is not tailored to the application, one model structure can be used for numerous applications, cf. e.g. [77] for a further discussion.

The problem of modeling nonlinear dynamical systems has not been as extensively covered as the modeling of linear systems. The main reason for this is that an introduction of nonlinearities greatly complicates the modeling procedure. For example, one linear model could be used to locally describe a large number of nonlinear systems. Consequently, if a linear model is not general enough to describe a particular nonlinear system there are several types of nonlinear models to choose from when determining a nonlinear model structure.

### 2.1.1 Grey-box Identification

A grey-box, or semi-physical, model is based on known relations of the system, and utilizes physical properties in e.g. mechanical processes, chemical reactions, or electrical circuits. The identification is focused on estimation of unknown parameters of the models, cf. [14; 34; 72]. The main advantage of this type of method is that available knowledge of the system dynamics is utilized in the modeling and optimization procedures. However, the use of first principles makes each model application specific and can therefore not be used for identification of a different type of system. This is particularly true in the nonlinear case. It also presupposes that the model structure derived from first principles is sufficiently complex to describe the system in question. There is also a risk of making the physical model too detailed, leading to an overly complex model structure and estimation of more parameters than necessary. In such cases model reduction may be of interest.

### 2.1.2 Black-box Identification

In cases when prior knowledge is limited or grey-box modeling for other reasons is difficult to perform, there are still a number of black-box approaches described in the literature that can be applied. This section gives an overview of nonlinear black-box identification methods. It should be noted that the methods mentioned here do not constitute a complete list but are merely examples of the wide range of methods for system identification that can be categorized as nonlinear and of black-box type.
Series Expansions

Some of the early black-box models were based on the Volterra series [76]

$$\hat{y}(t) = \sum_{k=1}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_k(\tau_1, \ldots, \tau_k) \prod_{i=1}^{k} u(t - \tau_i) d\tau_i$$  \hspace{1cm} (2.1)

where \( u(t) \) can be interpreted as an input signal, and \( \hat{y}(t) \) an output. From a system identification point of view the objective is to determine the Volterra kernels \( h_k(\tau_1, \ldots, \tau_k) \) for

$$\tau_i = 0, \tau_i^1, \ldots, \tau_i^{N_i}, \quad i = 1, \ldots, k$$  \hspace{1cm} (2.2)

where \( N_i \) is the number of points in which each \( \tau_i \) is evaluated, assuming a discrete time setup. It is also possible to use Volterra related methods in combination with frequency domain estimation methods as in [86].

Wiener used orthogonalization of the Volterra series to develop a different series expansion

$$\hat{y}(t) = \sum_{m=0}^{\infty} G_m(k_m, u(t)),$$  \hspace{1cm} (2.3)

see [76] and [90]. The advantage of the Wiener series over the Volterra series is that when the input is white Gaussian noise the Wiener functionals \( G_m(k_m; u(t)) \) are orthogonal. The Wiener kernels \( k_m(\sigma_1, \ldots, \sigma_m) \) can then easily be separated, which greatly facilitates the identification procedure. However, it should be mentioned that the large number of unknown parameters, also for low order systems, is a major drawback of these methods.

Block Oriented Methods

Another way of interpreting the Wiener model (2.3) is as block dynamics. The Wiener model can then be seen as a cascaded system consisting of a multiple input-multiple output (MIMO) linear dynamic system to which a static nonlinearity is applied at the output. The above observation is the reason why systems with the block structure of Fig. 2.1a are denoted Wiener systems. This block structure has a counterpart in the Hammerstein model where the static nonlinearity acts directly on the input signal and the linear block acts on the transformed input (see Fig. 2.1b). Though visibly similar, it is significantly easier to identify the linear dynamics in the Hammerstein model than in the Wiener model. The reason is that the Hammerstein model can be transformed to a linear multiple input-single output (MISO) model by a suitable choice of parameterization. A wide variety of methods have been developed based on the structures of Fig. 2.1, and related block structures. See e.g. [84; 89] and the references therein for detailed algorithms and analyses.

The NARMAX Class

A more general input-output model structure is provided by a nonlinear difference equations approach denoted NARMAX [30]. This model structure can be
seen as the nonlinear generalization of the linear ARMAX model. Put differently the ARMAX model (Auto Regressive Moving Average with eXogogeneous inputs)

\[ A(q^{-1})y(t) = B(q^{-1})u(t) + C(q^{-1})e(t) \]  

(2.4)
is a special case of the NARMAX (Nonlinear ARMAX) model

\[
y(t) = F(y(t-1),\ldots,y(t-n_y),u(t-1),\ldots,u(t-n_u),
\]
\[
e(t-1),\ldots,e(t-n_e)). \]

(2.5)
Here \( q^{-1} \) is the backward shift operator \( (q^{-k}y(t) = y(t-k)) \), \( y(t) \) the output, \( u(t) \) the input, and \( e(t) \) white noise. \( A(q^{-1}), B(q^{-1}), \text{and } C(q^{-1}) \) are polynomials and \( F(\cdot) \) is an arbitrary nonlinear function. As for NARMAX/ARMAX there are nonlinear counterparts to other linear model structures as well, e.g. NARX and NFIR correspond to ARX and FIR in the linear case [77].

The NARMAX model structure is very general, in fact, it is so general that unless the type of nonlinearity of the system is known, it may become difficult to choose a suitable function \( F(\cdot) \). As a polynomial can, at least locally, model any sufficiently smooth nonlinearity, it is suggested in [30] that \( F(\cdot) \) should preferably be chosen as a polynomial.

The difference equation sometimes has the advantage of enabling least squares techniques without requiring differentiation. The reason is that the differentiation is replaced by shifting. It should be noted that the NARMAX is a discrete time model, which sometimes requires conversion to continuous time. This is a complex task that sometimes causes problems [8].

**Neural Networks**

Closely connected to the NARMAX model is the neural network [31] which can be used for estimation of the function \( F(\cdot) \) of (2.5). Typically a neural network consists of multiple computational elements (nodes) arranged in layers. Each layer operates in parallel and each node is connected to all nodes in the adjacent layers (but not to the nodes in its own layer), see Fig. 2.2. Through
2.1 Nonlinear System Identification

Figure 2.2: Multilayer structure of a neural network. The nodes of each layers are connected to all the nodes of the adjacent layers. Through communication between the nodes weights that fit the model to the measured data are calculated.

communication between the nodes the network is then utilizing the information in the measured data to build a model of the system from which the measured data was obtained. The generality of neural networks make them useful primarily in cases when little or no information about the system dynamics is available. One major limitation of the neural network is that as the number of nodes increases the number of estimated parameters grows rapidly, which in turn requires very large data sets to obtain reliable results. As is well known, validation of the model needs to be carried out with care when neural networks are applied.

2.1.3 Modeling Approaches

Discrete and Continuous Time Models

Apart from choosing a model structure a decision needs to be made about whether the nonlinear model of the system should be formulated in discrete or continuous time. For discrete time systems the choice may be easy, but data from continuous time systems is usually collected through sampling, which means that the data is discrete. There is consequently a choice between a model which describes a discretized version of the system at certain sampling instances, and a model that describes the continuous time system. One obvious advantage of the use of a discrete time model is that it seems intuitive to fit discrete time data to a discrete time model. In addition, many of the methods
for system identification developed during the last three or four decades have been focused on discrete time modeling, so there are numerous tools to choose from. Other advantages include that it is easy to handle noise and time delays.

Continuous time models, on the other hand, provide a description of the continuous time system, which is particularly useful in controller design, as most nonlinear control theory is based on a continuous time description of the system. For grey-box models, which are common in continuous time modeling applications, there is the additional advantage of having physical interpretations of the parameters. Conversion of a discrete time model to continuous time can be complicated, particularly for nonlinear systems. If the sampling has generated non-minimum phase zeros, it may even be impossible, at least for the linear case [6]. The sampling and reconversion to continuous time may in itself introduce errors in the discrete time model. On the other hand, generating a continuous time model from sampled data will in many cases require calculation of numerical approximations of derivatives. For the derivative approximations to be accurate the sampling rate is required to be high, which in turn makes the the derivative approximation sensitive to noise.

For discrete time linear systems the Z-transform, and the corresponding shift operator $q$ is normally used, while it is common to use the Laplace transform and the differentiation operator $p$ when studying linear continuous time systems and models. The discrepancy between stability regions and the influence of non-minimum phase zeros during sampling can make it difficult to reconvert a model from discrete to continuous time after identifying the discrete time parameters. In [63] a different approach is suggested. It involves the $\Delta$-transform, with corresponding operator

$$
\delta = \frac{q - 1}{T_s},
$$

where $T_s$ is the sampling interval. This $\delta$ operator is the equivalent of a first order (Euler) approximation of a derivative, which is also used in this thesis. Consequently the properties of the $\delta$ operator and $\Delta$-transform approach those of the $p$ operator and Laplace transform as the sampling frequency increases. This implies that the effect of non-minimum phase zeros are reduced with increased sampling frequency, which is not the case for the Z-transform equivalent [63]. A drawback of the $\delta$ operator is an increased sensitivity to measurement noise as the signal is essentially numerically differentiated. This sensitivity is increased with high sampling frequencies, but can be counteracted through low pass filtering of the signal.

**Equation Error and Output Error Modeling**

Another choice that is relevant in this thesis is the one between equation error and output error (OE) modeling. To explain the differences, consider a linear dynamic system described by

$$
y(t) = \frac{B_0(q^{-1})}{F_0(q^{-1})} u(t) + w(t)
$$

(2.7)
where $y(t)$ is the measured output, $u(t)$ the input, and where $B_0(q^{-1})$ and $F_0(q^{-1})$ are polynomials in the backward shift operator $q^{-1}$. When an equation error model approach is used, the assumption on the noise $w(t)$ is expressed as

$$w(t) = \frac{1}{F_0(q^{-1})} e(t),$$

where $e(t)$ is white noise. This assumption is consistent with the following ARX model

$$y(t, \theta) = -\sum_{i=1}^{n_y} f_i y(t - i) + \sum_{i=1}^{n_u} b_i u(t - i) + e(t) = \varphi^T(t) \theta + w(t) \quad (2.9)$$

$$\varphi^T(t) = (-y(t-1) \ldots - y(t-n_y) u(t-1) \ldots u(t-n_u)) \quad (2.10)$$

$$\theta^T = (f_1 \ldots f_{n_y} b_1 \ldots b_{n_u}). \quad (2.11)$$

In particular it should be observed that the regression vector in this case consists of measured data.

In situations when the noise (or output error) of the model is not consistent with (2.9) the equation error model is not directly applicable. In such situations an output error model can be built up from the input and the parameters as

$$\hat{y}(t, \theta) = \frac{B(q^{-1}, \theta)}{F(q^{-1}, \theta)} u(t) \quad (2.12)$$

i.e.

$$\hat{y}(t, \theta) = -\sum_{i=1}^{n_y} f_i \hat{y}(t - i, \theta) + \sum_{i=1}^{n_u} b_i u(t - i) = \varphi^T(t, \theta) \theta \quad (2.13)$$

$$\varphi^T(t, \theta) = (-\hat{y}(t-1, \theta) \ldots - \hat{y}(t-n_y, \theta) u(t-1) \ldots u(t-n_u)) \quad (2.14)$$

$$\theta^T = (f_1 \ldots f_{n_y} b_1 \ldots b_{n_u}). \quad (2.15)$$

It is stressed that in (2.13), the regressor does not contain measured output data but simulated outputs, obtained from the estimated model.

**Least Squares and Prediction Error Methods**

In the equation error case the nonlinear prediction error criterion

$$V(\theta) = \frac{1}{N} \sum_{t=1}^{N} \epsilon^2(t, \theta), \quad (2.16)$$

where $\epsilon(t, \theta)$ is the prediction error and $N$ the number of data samples, collapses to the linear least squares criterion

$$V(\theta) = \frac{1}{N} \sum_{t=1}^{N} (y(t) - \varphi^T(t) \theta)^2. \quad (2.17)$$
Hence, equation error models are closely tied to an application of least squares (LS) and Kalman filter theory [58]. No such simplification exists in the output error case, which remains nonlinear due to the dependence of \( \theta \) in the regression vectors \( \varphi(t, \theta) \), i.e. the criterion is

\[
V(\theta) = \frac{1}{N} \sum_{t=1}^{N} (y(t) - \varphi^T(t, \theta)\theta)^2.
\]  

(2.18)

As a consequence, nonlinear iterative search algorithms [59] are generally required for the minimization of (2.18). Typical choices can be obtained by application of Gauss-Newton and gradient search directions [58].

**Consequences**

The equation error/LS and the output error/PEM type algorithms have some fundamentally different properties. First, the criterion of the equation error/LS type algorithm has a unique minimum point, provided that the excitation is sufficient and that the model is not over-parameterized [79]. Hence, under beneficial conditions, the algorithm always gives the same asymptotic result, regardless of the initialization. The criterion function of the output error/PEM type algorithm, on the other hand, may have multiple suboptimal minimum points to which the algorithm may converge.

It should be noted that the output error approach also have important advantages as compared to equation error models. The advantages include a better ability to handle unmodeled dynamics [87]. In particular, equation error methods tend to over-emphasize high frequency behavior since they are designed for prediction. The OE methods also have an on-line feedback from parameters to model signals that counteracts nearly unstable models within the algorithm, while LS methods sometimes give unstable models, even though the system may be stable. For the OE it is usually recommended to use a projection that prevents updating of the model parameters if they are about to leave the stability region. As the method is simulation based the lack of such a projection may lead to unstable model parameters and algorithm divergence. The equation error model is based on prediction and does hence not run the same risk of diverging. However, a stability check would need to be added to the algorithm anyway, to prevent complications related to unstable models. The ability to handle colored noise without obtaining biased estimates is another important robustness property where the OE method has an advantage compared to the equation error method [87].

### 2.1.4 Batch and Recursive Identification

All system identification methods can be characterized as being either recursive or non-recursive. In the latter case a whole batch of data is used to compute an off-line estimate of the model parameters. A recursive method, on the other hand, is performed with a gradual update of the parameter estimates, where the parameter estimate at time \( t \) is a function of the parameter estimate of the previous time step and the measured data obtained at time \( t \). A recursive
method can hence be applied on-line, with gradual addition of new measured data, or off-line. The main advantage of a recursive method over a non-recursive method is that it can be tuned to track changes of model parameter values over time, whereas the non-recursive methods lack this ability. The reference [58] discusses the main methods for design of recursive algorithms. Many system identification methods can be formulated both as recursive algorithms and as a non-recursive implementations. For prediction error algorithms the recursive and off-line methods asymptotically give equivalent results [58].

When the method is used off-line for a limited data set, and tuning of the algorithm fails to make it converge in the number of samples available, it is possible to apply multiple scans. Multiple scans means that the identification algorithm is applied to the data set iteratively, each time with an initial parameter vector that is equal to the final parameters of the previous scan. For obvious reasons the method is not applicable for on-line use, but could be used for a small data set off-line to provide initial parameters for on-line tracking or similar [80].

2.1.5 Analysis

There are numerous properties of identification methods which can be investigated through different types of analyses. Tracking abilities for recursive methods [37; 38], and accuracy analysis for estimation using the Cramér Rao Lower Bound (CRLB), see e.g. [79], are two examples. In this thesis the main analysis tool is averaging, which is used to investigate the parameter convergence properties of one of the identification methods.

Averaging

One important aspect when evaluating an identification method is the parameter convergence properties of the algorithm used. In [54] a method for analyzing the convergence of an identification algorithm

\[
\begin{align*}
x(t) &= x(t - 1) + \gamma(t)Q(t;x(t - 1),\varphi(t)) \\
\varphi(t) &= A(x(t - 1))\varphi(t - 1) + B(x(t - 1))e(t)
\end{align*}
\]  

(2.19)

for general linear systems was presented. Here \(x(t)\) are the estimates, \(\varphi(t)\) the observations and \(Q(\cdot;\cdot,\cdot)\) is a deterministic function. Further, \(e(t)\) is a sequence of random vectors, while the linear functions \(A(\cdot,\cdot)\) and \(B(\cdot)\) describe the updating of the observations. The idea is that by introducing a number of regularity conditions for (2.19), there is an associated ODE corresponding to the algorithm, and the convergence of the algorithm is linked to the stability of the ODE. Obviously, the analysis is only valid in the limit where the adaption gain \(\gamma(t) \to 0\). Hence, it is possible to analyze the stability of the ODE and from the results draw conclusions about the convergence of the algorithm. This is highly beneficial considering the many tools available for stability analysis of nonlinear systems.
In [53] similar results are discussed but for a more general nonlinear identification algorithm

\[ x(t) = x(t-1) + \gamma(t)Q(t; x(t-1), \varphi(t)) \]
\[ \varphi(t) = g(t; x(t-1), \varphi(t-1), e(t)). \]  \hspace{1cm} (2.20)

Here \( g(\cdot; \cdot, \cdot, \cdot) \) is a deterministic function that is restricted to fulfill

\[ \| g(x, \varphi, e) \| < C \quad \forall e, \varphi \quad \forall x \in D_R, \]  \hspace{1cm} (2.21)

where \( D_R \) is a subset of the \( x \) space where certain regularity conditions hold. Note that the main difference between (2.19) and (2.20) is the nonlinear generation of observations in the latter. This makes (2.20) suitable for analysis of many nonlinear identification algorithms, and provide a means of formulating conditions under which the algorithm can be expected to converge. Other frameworks than that of [53; 54] have also been used to derive associated ODEs. Typically stochastic differential equation theory is used, see e.g. [47].

2.2 Nonlinear Control

Controller design is, as for the identification problem, complicated by the introduction of nonlinearities. Which control schemes are applicable to a problem is to a large extent determined by the properties of the system. Therefore many methods require an accurate model of the system to enable proper design of the controller. Typically such models are written as general nonlinear ODEs on state space form [44; 65]. Unknown model parameters may then be obtained through nonlinear system identification.

2.2.1 Reusing Linear Control

Linearization

Through linearization of a nonlinear system, see e.g. [44], a linear approximation of the system can be obtained. A linear controller can thereafter be designed to fit the requirements for the linearized system. This method is easy to use. It can be expected to work particularly well for systems where the operating range of the controller is limited, and the system is nearly linear over the operating range. It is a standard tool which is normally the first to be applied when a new control system is designed.

Gain Scheduling

By linearizing a nonlinear system around an operating point it is straightforward to use linear control theory. However, the operating range of the controller will be limited to the range where the linearized system substitutes a good approximation of the nonlinear system. Gain scheduling [7; 44] utilizes the simplicity of the linearization/linear control design scheme, but instead of performing the linearization at a single operating point the system is linearized.
at a number of operating points with varying gain. For each operating point a linear controller can then be designed. Through the combination of the linear controllers a controller that meets the design requirements over a wider operating range can be obtained.

In tracking, gain scheduling can be expected to work reasonably well as long as the time varying scheduling variables have a significantly larger time constant than the system itself and the scheduling variables start close to the equilibrium. A major drawback of gain scheduling is that it only guarantees local properties of the controlled system. This implies that the reference value of the controller may have to be changed gradually through ramping or a sequence of smaller steps rather than by one large step in order to remain in the stability region. Despite this disadvantage, gain scheduling is an important technology, used e.g. in autopilots of commercial aircraft [3; 7].

**Adaptive Control**

As the name suggests the aim of adaptive control is to design a controller that adapts its behavior according to changes in the system dynamics of the process of interest. When controlling a nonlinear system with an adaptive controller using linear models, the controller hence adapts to the changing linear dynamics on-line. The adaptive controller can be regarded as consisting of two loops, a regular feedback loop, and a slower parameter adjustment loop, see Fig. 2.3. Though usually based on linear control theory, the controller becomes nonlinear through the parameter adjustment mechanism. Methods for adaptive control can be characterized as being either direct or indirect. The former represents design schemes where the controller parameters are adjusted without determining process and disturbance characteristics, something that is required in the latter. It should be noted that adaptive control can also be based on nonlinear methods, such as those in the next section.
Figure 2.4: Block structure of feedback linearization block structure for a Wiener system. Through feedback of the inverse of the static nonlinearity, \( f^{-1}() \), the closed loop system becomes linear, and the linear controller \( F(s) \) can be designed to control \( G(s) \). Further, \( y_{ref} \) is the reference signal for the output \( y \), and \( u \) is the input.

### 2.2.2 Nonlinear Feedback Control

Most control schemes include feedback in one form or another. Some examples of nonlinear feedback control methods include gain scheduling, feedback linearization, Lyapunov redesign, sliding mode control, nonlinear damping, and backstepping, some of which are discussed in this section. There is also the possibility of utilizing the knowledge of the system through model predictive control (MPC). A model of the system is then used for prediction of the expected effect of a certain control sequence.

Stability is a key property that is greatly complicated when nonlinear dynamics are introduced. It is stressed that stability analysis for control is not within the scope of this thesis, see e.g. [44] for treatment of this subject.

**Feedback Linearization**

Feedback linearization utilizes that for certain systems the nonlinearity can be eliminated by the right choice of nonlinear feedback [44]. The resulting closed loop system will then be perfectly (not just approximately) linear, which is advantageous as the control problem can thereafter be treated using standard linear control tools. A very simple example is shown in Fig. 2.4. This example does not describe the procedure for feedback linearization of general nonlinear systems. The purpose of the example is merely to illustrate the principle. A variable transformation of the states of the system is often necessary before the system can be linearized through feedback. There are several variations of feedback linearization; input-output linearization, full state linearization and state feedback control for partially linearizable systems. The applicability of feedback linearization is limited mainly by the structure of the system; far from all systems can be linearized through feedback, but when an option the method is powerful to use. It can be remarked that the method is based on an ODE model of the plant.
2.2. Nonlinear Control

Backstepping

Assume that a state feedback control law $u = \phi(x)$ is to be designed to stabilize a system of the form

$$\dot{x} = f_0(x) + g_0(x)z_1$$

$$\dot{z}_1 = f_1(x, z_1) + g_1(x, z_1)z_2$$

$$\vdots$$

$$\dot{z}_k = f_k(x, z_1, \ldots, z_k) + g_1(x, z_1, \ldots, z_k)u$$

as in [44]. The idea is to start by treating a part of the system (2.22) of interest. For this subsystem a feedback control signal is designed and a corresponding Lyapunov function found while treating $z_1$ as the control input. In the next step the first two equations are considered, which leads to a system of one degree higher order where $z_2$ can be treated as control input while $z_1$ is regarded as an additional state.

The recursive procedure where the system is gradually being extended leads up to the complete system (2.22)-(2.24) with $u$ as input, see [44]. Hence, the form of the system allows for the problem of finding a stabilizing control signal to be split into a number of smaller problems that are gradually made more complex until the full problem has been solved. This is a powerful control design tool. One limitation is that it requires the system to be written on the form (2.22)-(2.24), which is clearly not possible for all systems [44]. Again a requirement is the availability of an ODE model of the plant.

Optimal Feedback Control

The idea of optimal feedback control is to find a feedback control that minimizes an optimization criterion, e.g.

$$J(t_0, x(t_0)) = \min_u \int_{t_0}^{T} V(x(t), u(t))dt$$

subject to the system equations

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

Here $x(t)$ is the state vector, $u(t)$ the control signal, and $V(\cdot, \cdot)$ is a function that generates the optimization criterion, while $f(\cdot, \cdot, \cdot)$ are the dynamic equations describing the system. The Hamilton-Jacobi-Bellman (HJB) equation

$$\frac{\partial J}{\partial t} + \min_u \left( V(x, u) + \left( \frac{\partial}{\partial x} J(t, x) \right) f(x, u) \right) = 0$$

subject to the terminal constraints

$$J(T, x(T)) = 0$$

constitutes the solution to the optimal control problem [25]. The unknown function $V(t, x)$ is the cost incurred from starting in $x(t)$ and controlling the
system optimally from time $t$ to $T$. The HJB equation needs to be solved backwards in time, starting from $t = T$ and ending in $t = t_0$. In optimal feedback control the initial state $x(t_0)$ is obtained using measurements of the output at time $t_0$ [64]. More extended versions of the HJB equation exists, see [25; 64]. However, the HJB is difficult to solve, even numerically. The reason is the need to discretize states and inputs. For problems with high orders the amount of data and the processing time become extremely large. Evidently, also this control method requires an ODE model for its operation.

Model Predictive Control

Another way of utilizing knowledge about the system of interest while designing a control scheme is through model predictive control (MPC). The principle [51] dates back to the 1960’s when it was suggested as a way of avoiding the computational complexity of the HJB equation. As the name suggests the method is based on the use of a model of the system to consider the predicted effect of the control variable. The method allows for a sequence of the control variables to be calculated, as opposed to only the next sampling instance for other control design methods. Note however that the whole control sequence is recalculated between each sample, just like for methods where only one sample of the control signal is determined at a time. After the work of [73] and others, MPC is today established as a standard tool in process industry, see e.g [27].

Model predictive control is usually based on optimal control, with a similar optimization criterion that is solved for the control $u$ [60]. Through the calculation of the optimal control variable for several time instants, the method is particularly useful for systems with limitations on the states and/or the inputs, and for systems with time delays. The method is, however largely dependent on reliable and accurate ODE models of the system. This is one important motivation for the identification methods treated in this thesis.

The optimization procedure that aims at finding an optimal control sequence can be performed by any of a number of numerical optimization algorithms. The basic idea is to use the measured data up to time $t$, and a model of the system in an iterative search for the control sequence $(u(t+1) \ldots u(t+M))$ that minimizes a criterion function, typically on the form

$$V(u) = \sum_{s=1}^{P} (y_{ref}(t + s) - \hat{y}(t + s))^2 + \sum_{s=1}^{M} \rho(u(t + s))^2$$

(2.29)

where $V(u)$ is the loss function, $y_{ref}(t)$ is the reference signal for the predicted output $\hat{y}(t)$, and $\rho$ is the penalty on the control signal $u$. The prediction horizon $P$ is the number of samples the prediction of the criterion covers, while the control horizon $M$ is the length of the control sequence to be optimized. Consequently $M$ determines the size of the optimization problem, and hence the computational load and execution time when the controller is used online. Note that the equation (2.29) represents the single input single output (SISO) case, but the criterion can easily be generalized to the multiple input multiple output (MIMO) case. To introduce equality constraints on the inputs
and states is relatively straightforward, while the treatment of inequality constraints depend on the nature of the constraint itself, for details see e.g. [24]. As compared to the HJB that solves the feedback problem, MPC is far less complex; no discretization of states and input is e.g. required.

2.2.3 Nonlinear Feedforward Control

For systems with measurable disturbances it is common to use feedforward control in some form, often in combination with feedback. The methods described below are strict feedforward methods. First two general feedforward methods are described assuming a linear system. The generalization of these two methods to the nonlinear case are then discussed.

**Disturbance Decoupling - Linear Case**

The idea behind disturbance decoupling is to find a stabilizing controller to a given control problem, with the additional constraint that the transfer function from disturbance to measured output is zero (or at least close to zero). In Fig. 2.6 a schematic picture of a feedforward controller is shown. Here $y_{ref}$ is a reference signal for the output, $u$ is the control signal, $y$ the output, and $v$ a measurable disturbance, while the transfer functions $G$ and $H_v$ describe the system dynamics, and $F_r$ and $F_v$ are controllers. The signal $y$ can be expressed as

$$y = GF_r y_{ref} + (H_v + GF_v)v.$$  \hspace{1cm} (2.30)

By choosing $F_v$ such that $H_v + GF_v = 0$ the influence of the disturbance on the output may be decoupled. There are limitations to the applicability of the method, the time delay from the disturbance through the control signal must
be smaller than the time delay from the disturbance directly to the output, as there is otherwise no chance of fully compensating for it. Further there are conditions on the transfer function from the input to the output, e.g. it needs to be minimum-phase in the SISO case (similar conditions exist for the MIMO case) [82].

**Predict and Cancel - Linear Case**

If a disturbance of the system can be measured, and a reliable model of the system is available, the model can be used to calculate the expected influence of the disturbance on the measured output. A control can then be found to counteract the effect of the disturbance [82]. The method known as 'predict and cancel' does exactly what the name suggests, and is for obvious reasons only applicable when the disturbance is measurable.

**Reusing Linear Feedforward Control**

For nonlinear systems, the two above feedforward schemes can be reused by application of linearization, gain scheduling and adaptive feedforward control. This approximates the nonlinear problem with a linear one, exactly as in the case of feedback control. One difference is that the availability of an accurate model is more critical for feedforward than for feedback, hence adaptive feedforward control [7] is sometimes particularly attractive.

**Disturbance Decoupling - Nonlinear Case**

Though straightforward for linear systems, the method is complicated to use in the nonlinear case, the reason being the need to replace the transfer function by a corresponding quantity. Nonlinear decoupling problems are inherently complicated and typically depend on differential geometric tools, see e.g. [65] for further details. Note that the theory of [65] is based on nonlinear ODEs.
2.3. Time Delays

Predict and Cancel - Nonlinear Case

As opposed to disturbance decoupling a generalization to the nonlinear case appears to be uncomplicated for the predict and cancel methodology. In [83], LQG techniques are used to optimize the filters of a combined feedback and feedforward controller in the SISO case. This controller can be used to predict and reduce the disturbance impact in an optimal way, for linear systems with time invariant delays. This approach could easily be extended to the nonlinear case, and possibly also time varying delays.

Nonlinear Feedforward MPC

The MPC principle described in Section 2.2.2 is easily modified to a feedforward setting. The change is introduced by replacement of any \( \hat{y}(t + s) \) of (2.29) formed by measurements, by a corresponding quantity computed solely from disturbances and inputs by prediction. Provided that an accurate nonlinear model is available, the MPC setting hence becomes a general feedforward control tool. In the present thesis, this idea is applied for control of a more complex nonlinear system with input dependent time delays. Again it is noted that continuous time nonlinear dynamic models are needed for controller design.

2.3 Time Delays

Time delays occur in systems for various reasons, examples include transport delays in pipes and different industrial processes, and transmission delays in telecommunication systems.

For the general time delayed system

\[
\begin{align*}
\dot{x}(t) &= f(x(t), u(t - \tau_1)) + w(t) \\
y(t) &= h(x(t), u(t - \tau_2)) + e(t)
\end{align*}
\]  

(2.31)  

(2.32)

the input \( u(t) \) has an impact on the state vector \( x(t) \) and the output \( y(t) \) that is delayed by the times \( \tau_1 \) and \( \tau_2 \) respectively. Further, \( w(t) \) and \( e(t) \) are noise sequences.

2.3.1 Time Delays in System Identification

The problems involved in identifying time delayed systems include determining the time delay. For a system with varying time delay this can be done using different approaches, mainly depending on how the time delay behaves. For certain systems, like those involving flows in pipes, the delay depends on the flow itself. If the flow can be measured it may be possible to model the time delay as a function of the flow to be identified off line. For more complicated time delays a better solution may be to estimate the delay on-line.

Regardless of the behavior of the delay, it usually complicates identification as compared to identification of non-delayed system. As a start, time delays imply the need for storing of large number of data samples. This is particularly true for systems where the time delay is much larger than the sampling period.
The use of delayed data may also significantly increase the computational complexity of the identification method.

**Enhanced Parameter Vectors**

One method for identifying model parameters for linear systems with unknown time delays is to use a model

\[ A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t) + w(t) \]  

(2.33)

where \( y(t) \) is the output, \( u(t) \) the input and \( w(t) \) a zero mean noise sequence. \( A(q^{-1}) \) and \( B(q^{-1}) \) are polynomials in the backward shift operator. The parameter \( d \) represents the smallest value the delay takes, while \( B(q^{-1}) \) is chosen large enough to cover all the other possible values of \( \tau \) of (2.31)-(2.32) [46]. This approach works well as long as there is only a small variation in the time delay and the range of the time delay is known. For systems with large variations in \( \tau \) and fast sampling the method generates very large \( B \) polynomials, which in turn means that each estimated parameter will have a reduced accuracy and the computational load will be large. Related approaches could be used to solve nonlinear identification problems with delays. The model described in Section 3.1, which is used in the present thesis, would then be augmented with term consisting of delayed inputs, with the delay covering the time window of interest. This is not pursued in this thesis.

**Other Linear Approaches**

Further, some methods for identification of time delays involve estimation of models with different time delays and then choosing “the best one” according to some performance index [61]. For systems with fast sampling this implies a large number of models to estimate. Other approaches include identification of the delay as a rational transfer function [13] to reduce the number of computations required, or the use of a delta operator [63] for better conditioned models. None of these approaches appear to be directly applicable in the nonlinear context, though.

### 2.3.2 Time Delays in Control

Studying the Laplace transform of a time delayed signal \( f(t) \)

\[ \mathcal{L}(f(t-T)) = \int_0^\infty f(t-T)e^{-st}dt = e^{-stT}\mathcal{L}(f(t))(s) \]  

(2.34)

it can be seen that the time delay occurs as a factor \( e^{-stT} \) on the transform. From a Bode plot of the system this implies that the amplitude curve remains unchanged but the phase is shifted. The larger the phase shift compared to the desired cut off frequency, the more stability issues are likely to occur in controller design. There are different ways of handling the time delays in a system without getting problems with stability, two examples are by the use of a Smith controller or through optimal predictive control.
The Smith Controller

Consider a system $G$ with a time delay. If a controller $F$ is designed for $y = Gu$, without consideration taken to the time delay, stability issues like those described above would likely occur. In a Smith controller [8] $F$ is designed for the non-delayed system, while the time delay is compensated for separately by the use of expressions for $G$ (system without delay) and $Ge^{-sT}$ (system with time delay). In practice this means that an internal predictor inside the controller is used to generate the time delay system compensation $Ge^{-sT}$. Figure 2.7 illustrates the block structure of a Smith controller.

Optimal Predictive Control

Another approach to controlling systems with time delays is to use the framework of optimal control, and to use the measured inputs (and possibly measurable disturbances) to predict the effect of the inputs that have already been fed to the system, but whose effect on the output is yet to be seen. From this point it is possible to use the MPC framework to determine a predicted optimal control sequence, or to treat just one time step. The advantages of the MPC approach is the same here as for systems without time delays; the prediction horizon allows for the user to see the effect of decisions not yet taken. Further, the approach is directly applicable to the nonlinear case. However, if MPC is sensitive to modeling errors for cases without time delays, this is even more so for systems with delays. The reason is that the total number of steps for which the output needs to be predicted will be the sum of the delay and the non-delay prediction horizon, if the prediction is to provide similar information on the effect of a chosen control. If the model is not good the calculation of an optimal control sequence will be affected not only by the model errors appearing inside the controller, but also by a possible error in the initial states of the controller due to modeling errors appearing in the predicted effect of old inputs. Consequently, to handle (2.31)-(2.32) an accurate nonlinear ODE model of the system is desired.
2.4 Solar Energy for Heating and Cooling

In the last few years issues involving environmental problems and global heating have been brought to the agenda, and is now discussed not only by researchers, but by ordinary people and non-scientific media. The fossil fuels that have been used for energy generation purposes is not an endless resource and the side effects of the use of these fuels is becoming more and more visible. A solution for the future must involve other energy sources, and then preferably of kinds that are sustainable and clean.

The sun is the main energy source of our planet, and solar energy is abundant, clean and free for all. It therefore seems likely that a sustainable solution for future energy supply at least to some extent will utilize this energy supply. In order to improve the competitiveness of solar energy compared to other energy sources control strategies that enable a maximal utilization of the energy at hand are needed.

Solar energy can be converted into electricity through the use of photovoltaics (not treated in this thesis) or to heat through the use of solar collectors. The liquid heated in a solar collector can then be used to generate steam for electricity production, to desalinate seawater [28], be used directly for domestic hot water systems or to produce cooling when connected to an absorption machine as described in [52]. Such absorption machines require constant inlet temperatures to achieve an efficient cooling effect [48]. This is particularly challenging to achieve as the solar energy can vary, often with a high frequency, on days with partly cloudy weather. In fact the outlet temperature of a solar collector is effected by not only solar radiation, but ambient temperature, the temperature of the heat transporting liquid as it enters the collectors, the flow through the collector, and weather conditions. Most of these factors can be measured but not manipulated, a fact that opens up for the use of feedforward control schemes.

The cold production in the world is increasing with an increased demand of e.g. air conditioning, primarily in parts of the world with a warm climate. It is interesting to notice that many areas with high cooling demand also have an extensive supply of solar radiation. The use of solar radiation for cooling purposes therefore appears promising compared to e.g. the heating case where the demand and supply are differently distributed in time and space. Solar refrigeration plants are usually used in pharmaceutical, medical and food industries, where their main application is in the transportation of vaccines kept under constant refrigeration to hospitals and remote areas [1]. An application less exploited is the use of solar energy in air conditioning systems. One advantage of the use of solar radiation for cooling is that, apart from the energy source generally being more abundant in areas with a warm climate, as pointed out above, there is also a synchronization in time between solar energy supply and cooling demand, since cooling is in general needed more when the solar radiation is high. This is advantageous both from in daily and a seasonal perspective, and implies a lower need of hot water storage, compared to systems designed to utilize solar energy for heating. The difficulty of storing heat for a longer period of time is the main drawback of the ancient application of solar
domestic hot water systems, which are most favorably installed in areas where electricity is used for water heating, electric rates are high, usage is large and constant, and the climate is sunny [88].

There are different procedures for cold production which use solar radiation as the primary source (see e.g. [75]). In [12] many projects and technologies for solar cooling and refrigeration are reviewed. Analysis of solar air conditioning systems are available in [62] and [49]. Further, an overview on solar air conditioning in Europe is presented in [9], which surveyed and analyzed different solar powered cooling projects from an energetic, economical and technological point of view.

One of the most successful methods of solar energy based air conditioning is by means of an absorption machine, which produces cold water when hot water is injected into its generator. As the water is heated by the sun, this type of air conditioning system reduces conventional energy consumption compared to electric cooling. In addition, considering the problem of the midafternoon peak in electricity consumption due to electric chillers, the idea of solar cooling is intriguing from a demand perspective; the chilling demand to a significant extent coincides with the availability of solar radiation, as pointed out in [101].

### 2.4.1 Cooling by Absorption

Absorption refrigeration is based on the transport of energy via vaporization and condensation in two connected vessels. One of the vessels contains a liquid refrigerant, while the other vessel contains a solution of an absorbent and the refrigerant. During the absorption process, the refrigerant vapor is absorbed by the solution, causing the pressure in both vessels to drop. This in turn results in an increased vaporization of the refrigerant, which creates the cooling effect in the refrigerant vessel. As the process is normally exothermic (releasing energy), heat needs to be removed from the vessel for the process to retain its absorption capability. Meanwhile, this process constantly dilutes the absorbant/refrigerant solution. Eventually the solution will be saturated with refrigerant, which needs to be removed from the solution and returned to the refrigerant vessel before further absorption can occur. This is solved by the separation process.

The separation process is essentially a reversal of the process described above. As the separation process is endothermic (requires energy) heat is added to the solution, while evaporation will occur in the refrigerant vessel as surplus heat is transferred to the surroundings. Since the process requires an addition of heat, this is where the solar energy can be of use. Note that the absorption and separation processes cannot occur simultaneously in the same vessels. An absorption refrigeration cycle combines the two processes [81].

### 2.4.2 Solar Collector Dynamics

The development of reliable models of the solar collector dynamics are necessary for simulation as well as for control purposes, in particular for feedforward control. The availability of a good model can save time and money in controller design and tuning. Much of the work modeling solar collector dynamics have
been done using first principles. According to [28] the energy balance can be described by partial differential equations (PDEs) representing the change of temperature in the collectors. This has been done in a more detailed way [26; 33; 69; 74], where the pipe and fluid temperatures have been treated separately, and in a more simplified version, see e.g. [43], where only the liquid temperature is treated. The simpler model takes the form

\[
A \frac{\partial T}{\partial t}(t, x) + q(t) \frac{\partial T}{\partial x}(t, x) = c I(t)
\]

(2.35)

where \(A\) is the cross section area of the pipe, \(T(t, x)\) is the fluid temperature, \(q(t)\) is the flow through the collectors, \(c\) (constant or temperature dependent) is determined by qualities of the collector and heat transportation medium, and \(I(t)\) is the solar radiation. This model requires the identification of a parameter set that in general is obtained through an optimization process using experimental data. This type of model cannot be easily used in the synthesis of some controllers due to its complexity. In such cases more simple approaches, that still manage to represent the main characteristics of the process, are needed.

In [50] a method based on a neural networks that determines a black-box model of a solar collector during service is presented. The inputs considered are the solar radiation and the wind speed. The choice of inputs makes it very difficult to use the model for control purposes, as no manipulable variables, like inlet temperature or mass flow rate of the fluid, are considered. A simulator of a solar cooling plant is described in [100], where the solar collectors output temperature has been modeled based on physical properties of the system. The flow, input temperature and solar radiation have been considered as inputs. One main objective of that simulator is to apply hybrid control.

### 2.4.3 Control of the Solar Collector Outlet Temperature

As pointed out earlier most of the factors affecting the outlet temperature of a solar collector can be measured but not controlled. The main control variable when controlling the outlet temperature of solar collectors is the flow through the collectors. However, by manipulation of the flow the inherent time delays of the system change. In addition some of the time constants of the system are linked to the flow, so the system dynamics are also affected. In [28] a survey of simpler control approaches to the solar collector control problem, like PID, feedforward and cascade control schemes, was presented. In general, the attempts using simple PID control of solar collectors have failed to show a satisfactory behavior over larger operating ranges. The feedforward experiments were more successful; the static gain of the plant can be predicted using simple static models of the plant. There have also been experiments exploiting combinations of feedforward and PID controllers.

More advanced control schemes like MPC, adaptive control, gain scheduling, time delay compensation, and linear quadratic Gaussian control (LQG) is discussed in [29]. It is stated there that most of the MPC schemes used for control of distributed solar collector fields can be characterized as either adaptive, robust, or nonlinear, and include a feedforward term. A multivariable self-tuning multipredictor adaptive regulator proposed by [32] shows good results
for experiments with rapidly changing radiation, but at the cost of a relatively high complexity. Many of the adaptive control schemes discussed in [29] show oscillatory behavior when the reference or disturbances change rapidly. Gain scheduling has been used in numerous studies, one successful example being [42] where high order local ARX models are used to design linear controllers for different operating ranges. Stability is examined by experiments in the paper. Time delay compensation schemes for control of solar collectors include e.g. controllers of Smith predictor-like structure. Other references in [29] concern control using nonlinear MPC based on grey-box or black-box models of the collectors, traditional nonlinear control schemes like feedback linearization, robust control based on e.g. LQG or $H_\infty$ theory, fuzzy logic and neural network controllers.

The conclusions in [29] state that the control performance of model based controllers where plant nonlinearity is taken into account are superior to simple controllers of PID type. It is further stated that the "ideal" controller should be nonlinear and of high order, though the choice of method generally involves a trade-off between low complexity and high performance.

Previous work on the control of solar collector plants also include e.g. a Smith-predictor based MPC scheme [67]. That algorithm was originally proposed in [66]. More recently an adaptive algorithm was applied to the same plant, see [100].
An RPEM for Identification of Nonlinear Systems

Ordinary differential equations (ODEs) is a common way of describing the dynamics of various physical systems and processes. Many modern controller design methods are e.g. based on such models [44; 65]. This chapter presents a recursive prediction error algorithm for identification of systems described by nonlinear ODEs with a restricted black-box parameterization. The algorithm constitutes a starting point for the system identification related work of this thesis.

3.1 Model Structure

To describe the model that will be used in a large part of this thesis, first consider a general system of nonlinear ODEs

\[
\begin{pmatrix}
  x_1^{(1)} \\
  x_2^{(1)} \\
  \vdots \\
  x_n^{(1)}
\end{pmatrix} = \begin{pmatrix}
  f_1(x, u, \theta_1) \\
  f_2(x, u, \theta_2) \\
  \vdots \\
  f_n(x, u, \theta_n)
\end{pmatrix}
\] (3.1)

where the superscript \((\cdot)^{(j)}\) denotes the \(j^{th}\) time derivative. Further,

\[
x(t) = (x_1(t) \ x_2(t) \ \ldots \ x_n(t))^T
\] (3.2)

is the state vector,

\[
u(t) = (u_1(t) \ \ldots \ u_1^{(n_1)}(t) \ \ldots \ u_k(t) \ \ldots \ u_k^{(n_k)}(t))^T
\] (3.3)

is the input vector, \(\theta_i\) is a vector of unknown parameters, and \(f_1(x, u, \theta)\), \(\ldots, f_n(x, u, \theta)\) are arbitrary nonlinear functions. This model structure could
be used to describe a wide class of nonlinear systems. However, the number of degrees of freedom is quite high, and as it turns out, too high. This can be seen by consideration of the linear case, which becomes over-parameterized [97]. In [97] it is also shown that (3.1) can, in a local environment around a nonsingular point, be written as

\[
\begin{pmatrix}
  x_1^{(1)} \\
  \vdots \\
  x_{n-1}^{(1)} \\
  x_n^{(1)} \\
\end{pmatrix} =
\begin{pmatrix}
  x_2 \\
  \vdots \\
  x_n \\
  f(x, u, \theta) \\
\end{pmatrix}
\]  

(3.4)

for some function \( f(x, u, \theta) \), where \( x_1, \ldots, x_n \) of (3.4) denote new state variables, obtained from (3.2) by a transformation. Note that this requires that the original description (3.1) does not include derivatives of the inputs. By choosing a polynomial parameterization of \( f(x, u, \theta) \), i.e.

\[
f(x, u, \theta) = \sum_{i_1=0}^{I_{x_1}} \cdots \sum_{i_n=0}^{I_{x_n}} \sum_{i_{u_1}=0}^{I_{u_1}} \cdots \sum_{i_{u_k}=0}^{I_{u_k}} \sum_{i_{v_k}=0}^{I_{v_k}} \theta_{i_1, \ldots, i_n, i_{u_1}, \ldots, i_{u_k}, \ldots, i_{v_k}}
\]

(3.5)

\[
\varphi^T(x, u) \theta
\]

where

\[
\theta = \begin{pmatrix}
  \theta_{00} & \cdots & \theta_{0I_{x_1}} & \theta_{0010} & \cdots & \theta_{00I_{v_k}} & \cdots & \theta_{00I_{v_k}}0 \\
  \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
  \theta_{0I_{x_1}} & \cdots & \theta_{I_{x_1}I_{x_2}} & \cdots & \theta_{I_{x_1}I_{x_2},I_{u_1}, \ldots, I_{u_k}} \\
\end{pmatrix}^T
\]

(3.6)

\[
\varphi(x, u) = \begin{pmatrix}
  (x_1)^{i_{x_1}} \cdots (x_n)^{i_{x_n}} (u_1)^{i_{u_1}} \cdots (u_{n_1})^{i_{u_1}} \cdots (u_k)^{i_{u_k}} \cdots (u_{k_1})^{i_{u_k}} \\
  \cdots \\
  \cdots \\
\end{pmatrix}
\]

(3.7)

it can be ensured that a fairly general class of systems can be described, at least locally (cf. [97]). To clarify the polynomial form of \( \varphi(x, u) \) the following examples are useful

**Example 1.** Assume a SISO system. The model order is chosen to 1 (first order model), and the polynomial degree of the state and input are selected as 1 and 2 respectively. The obtained model structure then has six unknown parameters

\[
\theta = (\theta_{00} \ \theta_{01} \ \theta_{02} \ \theta_{10} \ \theta_{11} \ \theta_{12})^T
\]

(3.8)
3.2. Discretization

Corresponding to the regressor vector

$$\varphi(x, u) = (1 \ u \ u^2 \ x \ xu \ xu^2)^T.$$  \hspace{1cm} (3.9)

The model can be written on the form in (3.5), as

$$x^{(1)} = \theta_{00} + \theta_{01}u + \theta_{02}u^2 + \theta_{10}x + \theta_{11}xu + \theta_{12}xu^2$$  \hspace{1cm} (3.10)

Example 2. Assume that the system is of second order with one input. The polynomial degree of each state and input are selected as 1. The obtained model then has eight unknown parameters

$$\theta = (\theta_{000} \ \theta_{001} \ \theta_{010} \ \theta_{011} \ \theta_{100} \ \theta_{101} \ \theta_{110} \ \theta_{111})^T$$  \hspace{1cm} (3.11)

corresponding to the regressor vector

$$\varphi(x, u) = (1 \ u \ x_2 \ x_2u \ x_1 \ x_1u \ x_1x_2 \ x_1x_2u)^T.$$  \hspace{1cm} (3.12)

It should be noted that the model order and the polynomial degrees of each state and input can be regarded as tuning parameters. The number of inputs and outputs are determined by the system. It would, however, be possible to exclude e.g. an input or an output if this variable is found uninteresting for modeling purposes. Also note that with an increased number of inputs or states the number of parameters to estimate grows rapidly. It is then possible to limit the number of unknown parameters by using only some of the polynomial elements of $\varphi(x, u)$. Note, however, that the number of parameters is normally much lower than when e.g. Volterra series models are applied.

3.2 Discretization

In [97] the $(p \times 1)$ output vector is given by

$$y = C_m x$$  \hspace{1cm} (3.13)

where $C_m$ is a known constant matrix and $x$ is the state vector given by (3.4). A treatment of nonlinear output equations appear in [94; 95]. Throughout most of this thesis $C_m$ is assumed to be a unit vector such that

$$y = (1 \ 0 \ldots 0)x.$$  \hspace{1cm} (3.14)

However, the analysis in Chapter 5 uses the more general formulation associated with the output equation (3.13).

In order to be able to formulate an identification algorithm from the model (3.4),(3.14) it first needs to be discretized. Applying the Euler integration method to (3.4), and using the sampling period $T_s$ together with the relations (3.14) and (3.5) the following discrete time model is obtained in [97].

$$\begin{pmatrix} x_1(t + T_s, \theta) \\ \vdots \\ x_{n-1}(t + T_s, \theta) \\ x_n(t + T_s, \theta) \end{pmatrix} = \begin{pmatrix} x_1(t, \theta) \\ \vdots \\ x_{n-1}(t, \theta) \\ x_n(t, \theta) \end{pmatrix} + T_s \begin{pmatrix} x_2(t, \theta) \\ \vdots \\ x_n(t, \theta) \end{pmatrix} \varphi^T(x(t, \theta), u(t)) \theta$$  \hspace{1cm} (3.15)

$$y(t, \theta) = (1 \ 0 \ldots 0)x(t, \theta)$$  \hspace{1cm} (3.16)
3.3 Algorithm

One of the methods, which is used and analyzed in this thesis, is a previously
published RPEM algorithm of output error type. The algorithm, which has
been described in [94–97] is formed from the discretized model (3.15)-(3.16)
and includes on-line estimation of the covariance matrix of the measurement
disturbances. The construction of the algorithm follows the standard approach of [58]. The RPEM is given by

\[ \eta(t) = y_m(t) - y(t) \]

\[ \Lambda(t) = \Lambda(t - T_S) + \frac{\mu(t)}{t} (\eta(t)\eta^T(t) - \Lambda(t - T_S)) \]

\[ R(t) = R(t - T_S) + \frac{\mu(t)}{t} (\psi(t)\Lambda^{-1}(t)\psi^T(t) - R(t - T_S)) \]

\[ \hat{\theta}(t) = [\hat{\theta}(t - T_S) + \frac{\mu(t)}{t} \Lambda^{-1}(t)\psi^T(t)\hat{\eta}(t)]_{D_M} \]

\[ \varphi(t) = \left( \begin{array}{c}
\left( u_k^{(n_k)}(t) \right) I_{s_k}^{(s_{k-1})} u_k^{(n_k-1)}(t) \ldots \\
\vdots \\
\left( x_1(t) \right) I_{s_1}^{(s_{s-1})} \ldots (x_n(t)) I_{s_n}^{(s_{s-1})} (u_1(t)) I_{s_1} \ldots \ldots (u_k^{(n_k)}(t)) I_{s_k}^{(s_{s-1})}
\end{array} \right) \]

\[ \begin{pmatrix}
\begin{array}{c}
x_1(t + T_S) \\
\vdots \\
x_{n-1}(t + T_S) \\
x_n(t + T_S)
\end{array}
\end{pmatrix} =
\begin{pmatrix}
\begin{array}{c}
x_1(t) \\
\vdots \\
x_{n-1}(t) \\
x_n(t)
\end{array}
\end{pmatrix} + \alpha T_S
\begin{pmatrix}
\begin{array}{c}
x_2(t) \\
\vdots \\
x_n(t) \\
\varphi^T(t)\hat{\eta}(t)
\end{array}
\end{pmatrix} \]

\[ y(t + T_S) = (1 \ 0 \ \ldots \ 0)x(t + T_S) \]  

(3.17)

\[ \frac{d\varphi}{dx_i}(t) = \left( \begin{array}{c}
\left( u_k^{(n_k)}(t) \right) I_{s_k}^{(s_{s-1})} \\
\vdots \\
2x_i(t) + 2x_i(t)u_k^{(n_k)}(t) + \ldots
\end{array} \right), \quad i = 1, \ldots, n \]

\[ \frac{d\varphi}{dx}(t) = \left( \begin{array}{c}
\frac{d\varphi}{dx_1}(t) \\
\vdots \\
\frac{d\varphi}{dx_n}(t)
\end{array} \right) \]

\[ \frac{dx_i}{dt}(t + T_S) = \ldots = \frac{dx_{n-1}}{dt}(t + T_S) = \frac{dx_n}{dt}(t + T_S) \]
3.4 Scaling of the Sampling Period

\[
\begin{pmatrix}
\frac{dx_1}{dt}(t) \\
\vdots \\
\frac{dx_n}{dt}(t)
\end{pmatrix} + \alpha T_S \begin{pmatrix}
\frac{dx_2}{d\theta}(t) \\
\vdots \\
\frac{dx_n}{d\theta}(t)
\end{pmatrix} + \varphi^T(t) + \hat{\theta}(t) \left( \frac{dx}{dz}(t) \right) \left( \frac{dx}{d\theta}(t) \right)^T \left( \frac{dx}{d\theta}(t) \right)^T \left( \frac{dx}{d\theta}(t) \right)^T \left( \frac{dx}{d\theta}(t) \right)^T \left( \frac{dx}{d\theta}(t) \right)^T 
\]

\[
\psi(t + T_S) = \left( 1 \ 0 \ \ldots \ 0 \right) \frac{dx}{d\theta}(t + T_S)
\]

Here $\varepsilon(t)$ is the $(p \times 1)$ prediction error, $y_m(t)$ is the $(p \times 1)$ measured output vector, $\Lambda(t)$ is the $(p \times p)$ running estimate of the covariance matrix of the measurement disturbance, $\mu(t)/t$ is the gain sequence. In [97; 99] $\mu(t)$ of the gain sequence has been selected as

\[
\tilde{\mu}(t + T_S) = \mu_0 \tilde{\mu}(t) + 1 - \mu_0 \\
\mu(t) = \frac{t}{t + \mu_1 T_S} \tilde{\mu}(t),
\]

where $\mu_0$ and $\mu_1$ are time constants. This generation of the gain sequence is used throughout the thesis. Further, $R(t)$ is the $(d \times d)$ running estimate of the Hessian, and $\psi(t)$ is the $(d \times p)$ gradient of the output prediction with respect to the parameter vector. The gradient is determined by dynamic recursion, using the dynamics from the linearized state space model of the system. The set of parameter estimates that give stable models, $D_M$, is introduced to ensure model stability and is determined by linearization. Further, $\alpha$ is a scale factor applied to the sampling period. Aspects of the above parameters are treated in the next subsection, and in [96].

The main motivation for using this prediction error method is its generality, as the model structure (3.4),(3.14) enables modeling of a very wide range of systems. It is also an advantage that it is easy to reconstruct a continuous time model from the result of the RPEM algorithm. As compared to e.g. Volterra and Wiener series the number of parameters is kept small. Since the parameter vector is unaffected by the sampling the obtained model parameters can be directly interpreted in the original continuous time system. The main weakness of the method is, as discussed in Section 2.1.4, that the criterion function can have multiple suboptimal minima to which the algorithm may converge, depending on the initial parameters.

3.4 Scaling of the Sampling Period

The scale factor $\alpha$, which is applied to the sampling period, was introduced to compensate for differences in magnitude between estimated quantities. Other methods of scaling often require prior knowledge of the expected range of each estimated parameter, which may be difficult to determine in advance. The scaling of the sampling period addresses the problem of state vector components of different magnitudes. This is, in fact, a common reason why scaling is required in the first place [97]. The scaling of the sampling period leads to a slightly modified criterion. The reason for this is that as the scale factor of the
sampling period appears in the discrete time state update

\[
\begin{pmatrix}
  x_1(t + T_S) \\
  \vdots \\
  x_{n-1}(t + T_S) \\
  x_n(t + T_S)
\end{pmatrix} = \begin{pmatrix}
  x_1(t) \\
  \vdots \\
  x_{n-1}(t) \\
  x_n(t)
\end{pmatrix} + \alpha T_S \begin{pmatrix}
  x_2(t) \\
  \vdots \\
  x_n(t) \\
  \varphi^T(t)\hat{\theta}(t)
\end{pmatrix}.
\] (3.19)

It is shown in Theorem 2 of [97] that by applying a scale factor \( \alpha \) to the sampling period \( T_S \) the states of the scaled problem \( x_i(t, \theta) \), and the states of the original (rescaled) problem \( x_i^r(t, \theta^r) \) are related in the following way

\[
x_i(t, \theta) = \left( \frac{1}{\alpha} \right)^{i-1} x_i^r(t, \theta^r), \quad i = 1, \ldots, n.
\] (3.20)

From this relation it follows that each elements of the regressor vector will be scaled differently, depending on the occurrence of the different states and their polynomial degrees. As a consequence there is a scaled parameter vector that minimizes the scaled problem. It is further shown in Theorem 3 of [97] that the relation between the original parameters \( \theta^r \) and the scaled parameters \( \theta \) can be described by

\[
\theta^r = diag_{i=1}^{i=n} \left( \alpha^{n-i_12-2i_3-\ldots-(n-1)i_{z_1}} \right) \theta.
\] (3.21)

This means that it is trivial to retrieve the original parameters from the scaled ones, at any point of the identification run. As shown in [96], the condition number of the estimated Hessian can be improved by several orders of magnitude through the use of the scaling method. Note that in the rest of the thesis \( \hat{\theta} \) is used to denote the estimated parameters of the scaled problem that is treated by the algorithm, while the estimated parameters of the original problem, which are obtained from the scaled parameters at the end of the run, are denoted \( \hat{\theta}_{rescaled} \).
Part II

Contributions
Chapter 4

Using the Nonlinear RPEM

The objective of this chapter is to illustrate how the RPEM of Chapter 3 can be used for identification of nonlinear systems. In order to do so two examples will be used; an anaerobic digestion process and a domestic solar heating system. The anaerobic digestion data was generated by a first principles model with approximately 30 states. The use of the RPEM illustrates how a significantly simpler second order model structure with 8 unknown parameters to estimate is sufficient to describe the nonlinear dynamics of the process.

The data set from the domestic scale solar heating system is quite small for conventional use with a recursive identification method. The example is used to illustrate how the concept of multiple scans can facilitate convergence for small data sets. The two examples provide a motivation for the subsequent chapter on identification.

4.1 Anaerobic Digestion

With increased population and migration to urban areas the need for sustainable ways to manage the large amounts of waste produced increases. Some of the waste, e.g. domestic compost material, waste from food production, cattle and pig manure, and sludge from wastewater treatment, is of organic origin and is decomposed by microorganisms in nature. However, the amounts or concentrations associated with urban areas or large scale food production pose problems.

In a bioreactor, organic material can be decomposed by microorganisms in an anaerobic environment, producing methane (biogas) that can be used for fuel, and extracting nutrients for fertilization. As a bonus, the volume of waste used for landfill is reduced, and the amount of leachate from the landfill decreases.

To get an efficient digestion process the environment in the bioreactor needs to remain propitious for the microorganisms. This control problem is complicated by the sensitivity and complexity of the system as well as by the slow
dynamics, which make pilot scale experiments time consuming. A first principles based model of the process can to some extent replace pilot scale experiments. However, if the model is to be used for control purposes it is preferable that the model structure is as simple as possible without losing important aspects of the system dynamics. The anaerobic digestion process is nonlinear, which must also be considered in modeling and controller design. Examples of previous studies of nonlinear modeling and control of anaerobic digestion and biogas production include [2], [39], and [70]. In [2] an adaptive controller based on a simple model of anaerobic digestion is proposed, in [39] a procedure for model structure selection is considered, and [70] treats control design based on a grey-box model of a biogas tower reactor. Contrary to previous work, the present approach is to find a simple nonlinear black-box model for the process.

4.1.1 System Description

The experimental data used in this example (see Fig. 4.1) comes from simulations [40] using the IWA ADM1, an anaerobic digestion model presented by the International Water Association [41]. The model is of grey-box type, describing the concentration of a substantial number of chemical compounds as well as the amount of biomass in the reactor. Implemented as a set of differential and algebraic equations, the model consists of 26 dynamic state concentration variables, and 8 algebraic variables per reactor vessel or element. If implemented as differential equations only, the model has 32 dynamic concentration state variables. This rather complex model can be used for detailed studies of different aspects of the decomposition process.

Since the time constants of microbial growth in an anaerobic digestion reactor are large, the model offers an alternative to long and costly experiments in a pilot scale reactor. It also enables the input signal to be persistently exciting in terms of frequency and amplitude, which is crucial to the nonlinear identification, without having to consider the environmental problems of varying process performance, associated with physical experiments.

In the ADM1 simulation used in this example the microorganisms are fed with domestic waste, but the initial values are adjusted to digestion of excess waste water sludge. It takes a while for the process to stabilize after the change of conditions, and therefore the first 100 (out of a total of 8000) samples were not used for the identification experiments.

4.1.2 Experiments

Data properties

As in linear identification it is important to use persistently exciting inputs when collecting data, in order not to miss important information on the system. Hence, the input needs to have a varying frequency content. Furthermore, since a nonlinear relation between inputs and output is assumed, a variation of input amplitudes is required to identify the nonlinear dynamics [93]. In the experiments described in this example the input is chosen as a pseudo random binary sequence (PRBS) like signal with varying amplitudes, see Fig. 4.1.
Figure 4.1: Simulation data from the anaerobic digestion model (ADM1), propionate concentration (top), and substrate input (bottom).

The steps of various amplitudes provides information on the gain at different operating points of the system. The data consists of an input signal, which corresponds to the substrate volume added to the process ($\frac{\text{m}^3}{\text{D}1}$), and an output that corresponds to the propionate concentration in the reactor ($\frac{\text{kg COD}}{\text{m}^3}$). The sampling period is 6 hours.

To obtain better conditions for identification the data was scaled; the input signal with a factor $10^4$, and the output with a factor 10. However, in all plots the input and output (ADM1 generated as well as modeled) are in their original scale to facilitate a physical interpretation of the results.

Model Selection

Identification experiments with the biogas reactor data were performed using the model and algorithm described by equations (3.4)-(3.7), and (3.17), respectively. Step responses from the ADM1 indicate that the system needs to be modeled as 2nd order, or more. Hence, the approach was to make identification experiments on a basic second order model structure,

$$\theta = (\theta_{000} \, \theta_{001} \, \theta_{010} \, \theta_{011} \, \theta_{100} \, \theta_{101} \, \theta_{110} \, \theta_{111})^T$$

$$\varphi(x,u) = (1 \, u \, x_2 \, x_2 u \, x_1 \, x_1 u \, x_1 x_2 \, x_1 x_2 u)^T,$$

and then use it as reference when changing the polynomial degree of the input, $i_u$ (cf. (3.5)). For comparison the identification was also performed on a model
containing only the linear parameters of the second order system, \( i.e. \)
\[
\varphi(x, u) = (1 \ u \ x_2 \ x_1).
\]  

(4.3)

**Initialization and Algorithm Tuning - General Considerations**

When initializing the RPEM algorithm, the choice of initial parameters, \( \hat{\theta}(0) \), and the magnitude of the Hessian, \( R(0) \), are of great importance for the result of the identification. Choosing suitable initial values for the parameter vector is not trivial. Preferably the vector should be chosen close to the optimal values, or else the algorithm may produce unstable parameter estimates or converge to a local minimum of the criterion function. The size of the Hessian, \( R \), determines how fast the parameters change, and with a badly chosen \( R(0) \), the algorithm may take very long to converge (too large \( R(0) \)), or cause unstable parameter estimates (too small \( R(0) \)).

In the experiments the initial parameter vector, \( \hat{\theta}(0) \), was chosen to correspond to a stable linear system. Attempts to use a linear \( \hat{\theta}(0) \) with a pole placement that would give a similar step response as the nonlinear ADM1 generated data, resulted in convergence to the boundary of the stability region after a few iterations with the RPEM. A \( \hat{\theta}(0) \) with more stable poles gave better results.

In models with a large number of parameters, it takes longer before all parameters have responded to system dynamics and converged. It can then be necessary to adjust the parameter updating gain, \((\mu(t)/t)R^{-1}(t)\psi(t)\Lambda^{-1}(t)\) of (3.17). Since the parameter updating gain is proportional to \( \mu(t) \) and \( R^{-1} \) the convergence speed can be increased by increasing \( \mu(t) \) or reducing \( R \). However, if the parameter updating gain is chosen too large there is an increased risk of instability. This is well known, see \( e.g. \) [58]. For the experiments in this example the default value of the initial Hessian \( R(0) \) has been chosen as the identity matrix.

Another important tuning parameter is the scale factor of the sampling period (see Chapter 3) and [97]. If the states differ significantly in magnitude the identification problem will be badly conditioned, and numerical problems will occur. By scaling the sampling period of the model, the eigenvalues of the Hessian can be adjusted (cf. [96]), and convergence speed will be improved. This way parameter sets that would otherwise lead to divergence and instability may converge. Similarly, the gain can be tuned to obtain more favorable conditions for convergence.

Though not treated in this example, it is worth mentioning that other ways of increasing model complexity to catch unmodeled system dynamics could be to increase the model order, or the polynomial degree of the states. Both methods may, however, introduce convergence problems and increase the risk of instability. In general increasing the degree of the states poses more of a problem than increasing the degree of the input, since this has a more complicated influence on the nonlinear dynamics.
Table 4.1: Initialization values for the different nonlinear test cases. \( i_u \) is the polynomial degree of the input, \( \hat{\theta}_u(0) \), \( \hat{\theta}_{x_1}(0) \), and \( \hat{\theta}_{x_2}(0) \) are the initial values of the linear parameters. \( \alpha \) is the scale factor of the sampling time, the initial Hessian \( R(0) = r I_{n_\theta} \), where \( I_{n_\theta} \) is the \( n_\theta \times n_\theta \) identity matrix, and \( n_\theta \) is the number of elements in the parameter vector \( \theta \).

<table>
<thead>
<tr>
<th>( i_u )</th>
<th>( \hat{\theta}_u(0) )</th>
<th>( \hat{\theta}_{x_1}(0) )</th>
<th>( \hat{\theta}_{x_2}(0) )</th>
<th>( \alpha )</th>
<th>( r )</th>
<th>( n_\theta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>-1</td>
<td>-1.5</td>
<td>1</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>0.1</td>
<td>-1.5</td>
<td>-0.5</td>
<td>1</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>-1</td>
<td>-1.5</td>
<td>0.4</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td>-1</td>
<td>-1.5</td>
<td>0.4</td>
<td>5</td>
<td>20</td>
</tr>
</tbody>
</table>

Identification Results

Initialization of the different test cases were performed starting with all nonlinear parameter values set to zero. The linear parameters were then adjusted to obtain a stable parameter estimation that converged. For example, the basic nonlinear model with a first degree input (\( i_u = 1 \)) was initialized with

\[
\hat{\theta}(0) = (0, 0.1, -1, 0, -1.5, 0, 0, 0)^T. \tag{4.4}
\]

Comparison to (4.1)-(4.2) shows that the nonzero elements correspond to the \( u \), \( x_2 \), and \( x_1 \) parameters respectively. To obtain convergence the initial values of the Hessian needed adjustment. In Table 4.1 the initial values for the linear parameter elements, the sampling scale factor, and the initial Hessian are given for the different nonlinear model structures.

Figure 4.2 shows a close up of the results of the linear model experiment when the identified model is used for simulation. To highlight the difference in model performance the plots only cover the time between day 500 and 1500. It is clear that the linear model structure only manages to describe a fraction of the system dynamics. The basic nonlinear second order model (see Fig. 4.3) performs radically better, and explains most of the output variations. The parameter estimates converge smoothly, which can be seen in Fig. 4.4.

Figures 4.5, and 4.6, show the results of simulations with \( i_u = 2 \), and \( i_u = 4 \), respectively. The behavior with \( i_u = 3 \) lies somewhere between that of \( i_u = 2 \), and \( i_u = 4 \). All three alternatives manage to model the main dynamics, but fail somewhat to describe some of the overshoots that occur for large changes in input substrate. It is also worth mentioning that a higher degree of \( u \) implicates more parameters to estimate, and thereby an increased difficulty to get the algorithm to converge. With increased model complexity the algorithm becomes more sensitive to the choice of initial parameters as well as to scaling and gain. The 3\textsuperscript{rd} and 4\textsuperscript{th} degree models also show a tendency to oscillate.

For the biogas data the simulations shows high accuracy already at models with a low polynomial degree of the input signal. The improvement compared to the linear model is substantial. Model accuracy improves slightly as \( i_u \) increases, but the low degree model (with \( i_u = 1 \)) is sufficient to describe most
4. Using the Nonlinear RPEM

of the system dynamics. The parameters and corresponding regressor vector of this model is

\[
\hat{\theta}(7900T_s) = \begin{pmatrix} 0.2002 & 0.5491 & -0.9533 & 0.1546 \\ -1.3290 & 0.6239 & -0.0930 & 0.0423 \end{pmatrix}^T \quad (4.5)
\]

\[
\varphi(x, u) = (1 \quad u \quad x_2 \quad x_2u \quad x_1 \quad x_1u \quad x_1x_2 \quad x_1x_2u)^T. \quad (4.6)
\]

The model structure opens for identification using live data, which would be very complicated with a grey-box model with a large number of parameters. The more complex grey-box model is likely to be more difficult to tune, and stability issues may easily occur due to the large number of parameters. A low degree model also simplifies control design significantly.

4.1.3 Discussion

The identification experiments in this example show that part of an anaerobic digestion process can be modeled, obtaining good results, with as little as eight parameters. This relatively uncomplicated nonlinear model could greatly simplify the issue of controller design for the complex biological system in a biogas reactor. Note also that the identified continuous time model is directly applicable to most of the nonlinear controller design methods of [44].

The identification experiments showed that a linear model was insufficient to describe the system dynamics, whereas nonlinear models that describe most
Figure 4.3: ADM1 (solid) and simulated (dashed) propionate concentration for the 2nd order model with $i_{x_1} = 1$, $i_{x_2} = 1$, and $i_u = 1$.

Figure 4.4: Parameter estimates for the basic nonlinear 2nd order model, $i_{x_1} = 1$, $i_{x_2} = 1$, and $i_u = 1$.
Figure 4.5: ADM1 (solid) and simulated (dashed) propionate concentration for the 2nd order model with $i_{x_1} = 1$, $i_{x_2} = 1$, and $i_u = 2$.

Figure 4.6: ADM1 (solid) and simulated (dashed) propionate concentration for the 2nd order model with $i_{x_1} = 1$, $i_{x_2} = 1$, and $i_u = 4$. 
of the dynamics could be found for models with input polynomial degree of 1 to 4. The first degree input nonlinear model was very accurate, and contains few parameters compared to e.g. the grey-box model that was used to generate the data.

4.2 Solar Heating

As discussed in Chapter 2 there are several well known techniques for utilizing solar energy for different purposes. One of the oldest, and perhaps most intuitive is the use of solar heating of buildings. The second example used for illustration of the RPEM algorithm is a domestic solar heating system consisting of a solar collector, a heat storage and a fan. The example illustrates how recursive identification can be performed successfully even for small data sets.

4.2.1 System Description

The main component of a solar heating system is the solar collector. Depending on application and locality of the system the collector can differ in design, but common to all solar collectors is that they are constructed to absorb the solar radiation and transfer it to heat. The heat is then transported by some kind of medium to a heat storage. The medium can be air, but more common is to use a fluid, e.g. oil or a water and anti-freeze mixture.

The energy gain from the solar collector is determined by the balance between absorbed energy and heat loss to the environment. The heat loss increases with temperature and limits the maximum temperature attained from the system. To optimize the amount of extracted energy from the system it is therefore of interest to minimize the temperature loss from the collector. Hence, the ideal collector giving maximum temperatures and maximum useful power outputs is highly absorbent, well-insulated, and exposed to intense solar radiation.

The data [56] used for the example is from a small scale solar heating system, see Fig. 4.7. The collector, which has air as energy transportation medium, is connected to a heat storage filled with pebbles that retain the energy from the collector. There is also a fan, which controls the air flow between the collector and the heat storage. Data consist of measurement of solar radiation, \(u_1(t)\), fan switch state (on/off), \(u_2(t)\), and the temperature of the heat storage, \(y(t)\), during approximately two days, see Fig. 4.8. The sampling period is 10 minutes and the total number of samples is 296.

Several different off-line modeling approaches have previously been applied to this data set. In [57] a linear model, a neural network approach, and a grey-box model that involves reparameterization are presented.

The amount of data is limited, and for many nonlinear black-box identification methods 296 samples may not be sufficient to obtain an accurate model, particularly when a recursive algorithm is used. However, this example aims to show that the RPEM algorithm (3.17) can produce highly accurate models even with a small data set, by multiple scans of the algorithm over the data
4. Using the Nonlinear RPEM

Figure 4.7: A schematic picture of the solar heating system, with the measured variables radiation (solar radiation), $u_1(t)$, fan switch state, $u_2(t)$, and heat storage temperature, $y(t)$.

Figure 4.8: Measured data from the solar heating system, heat storage temperature, $y(t)$ (top), solar radiation, $u_1(t)$ (middle), and fan switch state, $u_2(t)$ (bottom).
set. Note that only 50% of the data is used for identification, the second half of the data set is used for validation.

4.2.2 Experiments

Application and Data

As mentioned previously the data from the solar heating system consists of two inputs and one output. The solar radiation $u_1$ varies over the day, and during the night there are, naturally, long periods of no solar radiation. This introduces some difficulties, since there is no excitation of $u_1$ during this time. The second output $u_2$ corresponds to the speed of the fan transporting warm air from the solar collector to the heat storage. The fan has one speed only, hence $u_2$ is a binary signal where 1/0 means on/off. Thirdly, the output signal $y$ represents the air temperature in the heat storage, measured at the inlet from the solar collector.

To be able to compare model performances the data set was split in two, one subset for calibration of the model parameters and a second for validation of the calibrated model. The calibration set consists of the first half of the samples and the validation data of the second half.

Small Data Set - Multiple Scans

As mentioned earlier, the data set used for calibration only contains 148 samples. From a modeling perspective this is a small data set, especially when considering that the identification method is of recursive black-box type. In fact, 148 data points is unlikely to be enough for the algorithm to converge from an arbitrary set of initial parameters. To avoid this problem multiple scans can be used.

The idea behind the multiple scans is to apply the algorithm to the data set several times, each time changing the initial parameters to the final parameters of the previous iteration. This idea has also been used in [80]. The adaptive gain is also reduced by applying the same procedure to the estimated Hessian. This way an arbitrary set of initial parameters could eventually converge to a minimum, even for a small data set. An acceptable model could be obtained from a small amount of data even if the initial guess is poor. If the data set would have been larger the multiple scans would be superfluous.

The nonlinear model structure increases the risk of instability issues and convergence to local minima significantly. There is no stringent way of determining if a model is in fact the best model obtainable, or if it is just a local minimum. Similarly, in case the algorithm fails to converge to a good model, it can be difficult to know whether this is because it lacks the required polynomial elements, or if it is a result of e.g. badly chosen initial parameters. However, it may not be crucial to find the optimal model, but rather one with a model performance satisfactory for the intended application.
Identification Results

Identification experiments with the solar heating system data have been performed using the model described by equations (3.4)-(3.7) through the RPEM (3.17). The experiments were of black-box type, where no prior knowledge of the system was assumed to be available. The starting point was then to use the complete $\varphi(x,u)$ and $\theta$ vectors

$$\varphi(x,u) = (1 \ u_2 \ u_1 u_2 \ x \ xu_2 \ xu_1 \ xu_1 u_2)^T$$

and then exclude one polynomial element at a time to determine which parameters improve the model performance the most.

The algorithm was initialized with

$$\hat{\theta}(0) = (0 \ 5 \ 0.5 \ 0 \ -0.08 \ 0 \ 0 \ 0)^T$$

A comparison to (4.7)-(4.8) shows that all the initial nonlinear parameters have been chosen to zero. The first element, corresponding to the bias parameter, was also chosen to be zero. The initial bias parameter did, however, cause difficulties initially. The estimation of the bias parameter totally dominated the identification procedure, leaving the static gains (element 2 and 3) relatively unchanged. This lead to convergence to a local minimum that resulted in a poor model. To solve this problem the bias parameter of $\hat{\theta}(0)$ was excluded until the static gains had adapted, after which it was added to the parameter vector again. The model obtained by this procedure was significantly better than the one obtained when just using (4.9).

Each of the other, reduced, models were obtained by removing one parameter at the time from the full eight parameter model. For some of the parameter combinations the estimates changed a lot over the calibration data set, resulting in bad models. The relatively large change of the parameter estimates, see Fig. 4.9, indicated that the parameter updating gain,

$$\frac{\mu(t)}{t} R^{-1}(t) \psi(t) \Lambda^{-1}(t),$$

was too high. This can be avoided by increasing the initial value of the Hessian, $R$, or reducing the magnitude of the gain sequence $(\mu(t)/t)$ as discussed in Section 4.1.2.

Examples of simulations with models obtained in the experiments are shown in Fig. 4.10 and 4.11. In Table 4.2 the mean square error

$$MSE = \frac{1}{N} \sum_t (y_m(t) - y(t))^2$$

of the simulated output with each model are presented.

The model performance was only improved when one of the parameters $\theta_{011}$ and $\theta_{110}$, was removed, all other parameter removals resulted in a higher MSE after convergence. (With $\theta_{010}$ removed, however, convergence was obtained after increasing the Hessian by a factor 3. With the original Hessian
Table 4.2: Experiments. The left column indicates which parameters are not included in the model. The right column contains the mean square error after convergence.

<table>
<thead>
<tr>
<th>Excluded parameters</th>
<th>Mean Square Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>None (full model)</td>
<td>15.0</td>
</tr>
<tr>
<td>\theta_{000}</td>
<td>17.0</td>
</tr>
<tr>
<td>\theta_{001}</td>
<td>15.5</td>
</tr>
<tr>
<td>\theta_{010}</td>
<td>15.7</td>
</tr>
<tr>
<td>\theta_{011}</td>
<td>14.7</td>
</tr>
<tr>
<td>\theta_{100}</td>
<td>15.1</td>
</tr>
<tr>
<td>\theta_{101}</td>
<td>19.1</td>
</tr>
<tr>
<td>\theta_{110}</td>
<td>14.6</td>
</tr>
<tr>
<td>\theta_{111}</td>
<td>15.6</td>
</tr>
<tr>
<td>\theta_{010},\theta_{011}</td>
<td>11.2</td>
</tr>
<tr>
<td>\theta_{010},\theta_{110}</td>
<td>12.1</td>
</tr>
<tr>
<td>\theta_{011},\theta_{110}</td>
<td>11.3</td>
</tr>
<tr>
<td>\theta_{010},\theta_{011},\theta_{110}</td>
<td>11.9</td>
</tr>
<tr>
<td>\theta_{010},\theta_{110},\theta_{111}</td>
<td>11.6</td>
</tr>
<tr>
<td>\theta_{010},\theta_{011},\theta_{100},\theta_{110}</td>
<td>46.3</td>
</tr>
</tbody>
</table>

the algorithm reached a point where it alternated between scans between two parameter estimates, one giving an MSE of 20.1 and the other 13.4.) The lower MSE suggests that removal of \theta_{010} may lead to improved model performance, even though the converged parameter set did not indicate that. The experiment was therefore extended to removing also different combinations of \theta_{010}, \theta_{011}, and \theta_{110}. As Table 4.2 shows, removal of any combination of two of these three parameters improved the model significantly. Removal of all three parameters also resulted in a relatively good model.

The original removal of only \theta_{100} did not seem to affect the model behavior significantly. However, an attempt to remove the \theta_{100} parameter from the model where \theta_{010}, \theta_{011} and \theta_{110} had been removed resulted in a very poor model.

As an extra experiment \theta_{011} was exchanged for \theta_{111} (in the model with three removed parameters) under the hypothesis that the important aspect would be to have a term in the model containing the product of the two inputs, but that it would be of less importance whether it was \(u_1u_2\) or \(xu_1u_2\). The hypothesis was supported by the identification results, the MSE was even slightly lower for the model where \theta_{111} was removed, compared to that where \theta_{011} was removed.

Finally, a case when the whole data set was used for identification is displayed in Fig. 4.12. The simulation was also performed over the entire data...
set, using the final parameter estimate from the identification.

**Comparison with an Alternative Model**

Since the same data set has been treated previously by [57] a comparison of model performance would be interesting. Hence, the off-line reparameterization experiment presented by Ljung and Glad has been performed on the same calibration and validation data sets to enable a comparison. The reparameterized model,

\[ y(t) = \theta^T(t) \varphi(t) \]
\[ \varphi(t) = (\varphi_1(t) \, \varphi_2(t) \, \ldots \, \varphi_6(t))^T \]  

(4.12)

where

\[ \varphi_1(t) = y(t - T_S) \]
\[ \varphi_2(t) = y(t - T_S)u_2(t - T_S)/u_2(t - 2T_S) \]
\[ \varphi_3(t) = y(t - 2T_S)u_2(t - T_S)/u_2(t - 2T_S) \]
\[ \varphi_4(t) = u_2(t - T_S)u_1(t - 2T_S) \]
\[ \varphi_5(t) = y(t - T_S)u_2(t - T_S) \]
\[ \varphi_6(t) = u_2(t - T_S)y(t - 2T_S) \]  

(4.13)
Figure 4.10: Measured (solid) and simulated (dashed) heat storage temperature for the full black-box model.

Figure 4.11: Measured (solid) and simulated (dashed) heat storage temperature for the model where $\theta_{010}$ and $\theta_{011}$ has been excluded.
is based on an energy balance of the solar heating system.

As in the RPEM model the off-line model has parameters that are more crucial to the model performance than others. In both cases there were also a substantial improvement when some of the more superfluous parameters were excluded from the model. Hence, the best model obtained with the RPEM model has been compared to a reduced version of (4.12)-(4.13), containing the parameters $\varphi_1(t), \varphi_2(t), \varphi_3(t), \varphi_4(t)$, and $\varphi_6(t)$. The reparameterization method also requires removal of background temperature before the identification event. In this case the background temperature has been assumed to be $21^\circ C$. It might have been possible to reduce the MSE of the off-line model further by subtracting another background temperature (perhaps time varying) or by using a different subset of parameters. The results appear in Fig. 4.13.

Some comments are in order here. Note that the model (4.12)-(4.13) settles to a constant output during periods with zero solar radiation. This is because only the product of $u_1$ and $u_2$ affects the output signal. A consequence is that this model underestimates the temperature during night. The black-box RPEM, however, retains a direct impact from the fan ($u_2$) on the output, a fact which allows the model to compensate somewhat during periods of zero solar radiation. This is why the output of the RPEM model does not settle to a constant value during periods when $u_1 = 0$.

The MSE of the RPEM and reparameterization methods were 11.2 and 11.9, respectively.
4.2.3 Discussion

The identification experiments performed in this example show that the dynamics of a small scale solar heating system can be identified with a recursive black-box model even for a very small amount of data, using multiple scans. The simple model structure and the general approach in addition to the small amount of data required opens up for e.g. improved controller design and thereby a more efficient utilization of the solar energy.

The identification experiments showed that the general black-box state space model provides a suitable starting point for identification of the nonlinear dynamics of the system. Through gradual removal of parameters, the model performance was significantly improved. The best model obtained was also compared to an energy balance based least squares method. The models proved to have comparable performance. Some differences of the models that are worth noting are that the RPEM method studied in this example is an OE method, which is known to perform better than equation error methods in simulations. Hence, it may be difficult to conclude which of the two approaches is better. Another thing worth mentioning is that due to the chosen discretization method the results from the RPEM can be interpreted directly in terms of the physical meaning of the continuous time parameters.
4.3 Summary

As shown by the two examples of this chapter the RPEM presented in Chapter 3 can be successfully used for black-box identification of different types of nonlinear systems. The anaerobic digestion example illustrates how the general nonlinear model structure can provide a powerful alternative to complex semi-physical modeling tools; the resulting model is considerably less complex without significant loss in performance (for the considered input signal). This result is particularly useful from a controller design perspective, and it directly enables the use of the nonlinear controller design methods of [44; 65] to this type of plant. The solar heating example shows that for identification experiments where only small data set are available, the concept of multiple scans [80] can be applied in an off-line environment, also when nonlinear dynamics are identified. Thereby the problems associated with not reaching convergence from arbitrary initial parameters can be circumvented. In addition, the two examples introduce techniques for systematic selection of terms in the model (3.5), and provides advice for tuning of the RPEM (3.17).

Despite the good final results of the identification experiments, the examples also show some weak points of the RPEM algorithm. Particularly the problem of tuning the algorithm in a suitable way to obtain convergence, and choosing initial parameters appear to be issues that require further attention. In the next two chapters these issues are addressed more in detail.
Convergence Analysis

As discussed in Chapter 4 the tuning of the RPEM algorithm of Chapter 3 greatly affects its convergence behavior. In [54] a method for analyzing the convergence of an identification algorithm for general linear systems is presented. The idea is that provided that a number of conditions are fulfilled, there is an associated ordinary differential equation (ODE) corresponding to the algorithm, and the convergence of the algorithm is linked to the stability of the ODE. Hence, it is possible to analyze the stability of the ODE and from the results draw conclusions about the convergence of the algorithm. In [53] similar results are discussed but for a more general nonlinear identification algorithm.

In this chapter a convergence analysis of the RPEM of Chapter 3 is performed, using the theory of [54] and [53]. The objective of the analysis is to formulate conditions which imply that convergence to a minimum of the criterion function of the RPEM algorithm (3.17) is possible.

5.1 Model Structure

As in the previous chapter the algorithm utilizes the state space model structure

\[
\begin{pmatrix}
  x_{1}^{(1)} \\
  \vdots \\
  x_{n-1}^{(1)} \\
  x_{n}^{(1)}
\end{pmatrix}
= \begin{pmatrix}
  x_{2} \\
  \vdots \\
  x_{n} \\
  \phi^T(x, u)\theta
\end{pmatrix},
\]

(5.1)

where \( x \), \( u \), \( \phi(x, u) \), and \( \theta \) are defined as in Chapter 3. Note that as the analysis focuses on the algorithm (3.17), the nonlinearity will be considered to be of polynomial form, as before. The output equation is of the form

\[
y(t) = C_m x(t),
\]

(5.2)

where \( C_m \) is a known constant matrix describing the relation between the state and output vectors. The model (5.1)-(5.2) is discretized using Euler integration,
as described in (3.15)-(3.16). The system is specified in more general terms than (5.1) and (5.2) in the conditions of the analysis. This makes the analysis valid also when the system is not contained in the model set.

5.2 Algorithm Modifications

To enable application of the theory from [53] a few minor modifications have been made to the algorithm (3.17). None of these modifications have any practical implication on the use of the RPEM algorithm, but are added for analysis purposes. First, one additional backward shift in time in the $R$ and $\theta$ recursions are introduced, thereby making the recursive algorithm fit the framework of Section 5.3. As the gain sequence is decreasing, and eventually approaches 0, there will be no asymptotic impact of the introduced time shift.

A second modification of the algorithm is to include the Hessian $R$, and the estimated measurement noise covariance matrix $\Lambda$ in the projection into the compact model subset $D_M \in \mathbb{R}^{d+\bar{a}^2+p^2}$. $D_M$ now defines the allowed values not only of $\theta$ (as in [97]), but also of $R$, and $\Lambda$. The projection is assumed to be given by

$$H(t) = (h(t))_{D_M} = \begin{cases} h(t) & h(t) \in D_M \\ H(t-1) & h(t) \notin D_M \end{cases}$$

$$H(0) \in D_M$$

(5.3)
as suggested in [58]. Here $H(t)$ represents the left-hand side of the recursions, while $h(t)$ is the corresponding right-hand side. Since $H(0) \in D_M$, it is guaranteed that $H(t) \in D_M$, $\forall t$. In general $D_M$ only contains $\theta$ that correspond to asymptotically stable time invariant linearized dynamics, and positive definite symmetric $R$ and $\Lambda$, i.e. $D_M = \{ \theta, R, \Lambda : |\lambda_i(S(\theta))| \leq 1 - \delta \ \forall i, \ R > \delta R > 0, \ \Lambda > \delta \Lambda > 0, \ \delta > 0 \}$, where $S(\theta)$ is the system matrix of the linearized system.

The third modification introduces a saturation $( )_{\text{sat}}$ in the recursions of $x(t)$ and $\frac{dx}{dt}(t)$. This, together with corresponding assumptions on the data generation, guarantees that the model and system are bounded. This would otherwise not have been the case, considering the polynomial of (3.5). Note that the saturation can be chosen arbitrarily large, and does therefore not affect the algorithm in practice. This follows since the model is assumed to be exponentially stable and bounded in $D_M$, see Appendix A for details. Finally the corners of the saturation are assumed to be smooth, so that differentiability conditions are not affected.

With these modifications the algorithm takes the form

$$\varepsilon(t) = y_m(t) - y(t)$$

$$\Lambda(t) = (\Lambda(t - T_S) + \frac{\mu(t)}{t} (\varepsilon(t)\varepsilon^T(t) - \Lambda(t - T_S)))_{D_M}$$

$$R(t) = (R(t - T_S) + \frac{\mu(t)}{t} (\psi(t)\Lambda^{-1}(t - T_S)\psi^T(t) - R(t - T_S)))_{D_M}$$
5.2. Algorithm Modifications

\[ \hat{\theta}(t) = (\hat{\theta}(t - T_S) + \frac{\mu(t)}{t} R^{-1}(t - T_S) \psi(t) \Lambda^{-1}(t - T_S) \varepsilon(t)) D_M \]

\[ \varphi(t) = \left( 1 \ldots \left( u_k^{(\eta_k)}(t) \right)^I_{\tau_k} t \right) \ldots \]

\[ \ldots \left( u_k^{(\eta_k-1)}(t) \right) \left( u_k^{(\eta_k)}(t) \right)^I_{\tau_k} \ldots \left( u_k^{(\eta_k-1)}(t) \right)^I_{\tau_k} \ldots \]

\[ \ldots \left( u_k^{(\eta_k-1)}(t) \right) \left( u_k^{(\eta_k)}(t) \right)^I_{\tau_k} \ldots \left( u_k^{(\eta_k-1)}(t) \right)^I_{\tau_k} \ldots \]

\[ \ldots \left( x_1(t) \right)^I_{\tau_1} \ldots \left( x_n(t) \right)^I_{\tau_n} \left( u_1(t) \right)^I_{\tau_1} \ldots \ldots \left( u_k^{(\eta_k)}(t) \right)^I_{\tau_k} \right) \]

\[ \left( \begin{array}{c}
  x_1(t + T_S) \\
  \vdots \\
  x_{n-1}(t + T_S) \\
  x_n(t + T_S)
\end{array} \right) = \left( \begin{array}{c}
  x_1(t) \\
  \vdots \\
  x_{n-1}(t) \\
  x_n(t)
\end{array} \right) + \alpha T_S \left( \begin{array}{c}
  x_2(t) \\
  \vdots \\
  x_n(t) \\
  \varphi^T(t) \hat{\theta}(t)
\end{array} \right)_{sat} \]

\[ y(t + T_S) = C_m x(t + T_S) \]  

\[ \frac{d\varphi}{dx_i}(t) = \left( \begin{array}{c}
  \frac{d\varphi}{dx_1}(t) \ldots \frac{d\varphi}{dx_n}(t) \end{array} \right)^T 
\]

\[ \frac{dx}{d\theta}(t + T_S) = \left( \begin{array}{c}
  \frac{dx_1}{d\theta}(t + T_S) \\
  \vdots \\
  \frac{dx_{n-1}}{d\theta}(t + T_S) \\
  \frac{dx_n}{d\theta}(t + T_S)
\end{array} \right) = \]

\[ \left( \begin{array}{c}
  \frac{dx_1}{d\theta}(t) \\
  \vdots \\
  \frac{dx_{n-1}}{d\theta}(t) \\
  \frac{dx_n}{d\theta}(t)
\end{array} \right) + \alpha T_S \left( \begin{array}{c}
  \frac{dx_2}{d\theta}(t) \\
  \vdots \\
  \frac{dx_n}{d\theta}(t) \\
  \varphi^T(t) + \hat{\theta}^T(t) \left( \frac{d\varphi}{dx}(t) \left( \frac{dx}{d\theta}(t) \right) \right)_{sat}
\end{array} \right) 
\]

\[ \psi(t + T_S) = C_m \frac{dx}{d\theta}(t + T_S). \]

As before \( y_m \) is the measured output, \( \varepsilon(t) \) is the prediction error, \( \Lambda(t) \) is the running estimate of the covariance matrix of the measurement noise, \( \mu(t)/t \) is the gain sequence, \( R(t) \) is the running estimate of the Hessian, and \( \psi(t) \) is the gradient of the output prediction. The subscript \( D_M \) denotes the projection into the compact subset \( D_M \in \mathbb{R}^{d + d^2 + p^2} \) defining the allowed values of \( \theta, R, \) and \( \Lambda \). The parameter \( \alpha \) is the scale factor of the sampling period as described in Chapter 3.
5.3 Associated ODE Analysis Tools

In the rest of this chapter a change of time notation is applied. Hence $t + T_s$ is replaced by $t + 1$ and similar. The subscript $g$ is introduced for $x$ and $\varphi$ variables of the general analysis tools of [54] and [53] to avoid confusion with the RPEM variables of (5.4).

The convergence properties of the recursive identification algorithm (5.4) is analyzed using the associated differential equation approach described in [54] and [53]. Ljung treats the general nonlinear algorithm

$$x_g(t) = x_g(t - 1) + \gamma(t)Q(t; x_g(t - 1), \varphi_g(t))$$
$$\varphi_g(t) = g(t; x_g(t - 1), \varphi(t - 1), e(t))$$

(5.5)

where $x_g(\cdot)$ are the estimates, and $\varphi_g(\cdot)$ the observations. Further, $Q(\cdot, \cdot, \cdot)$ is a deterministic function corresponding to the updating direction, while $g(\cdot, \cdot, \cdot)$ is a deterministic functions describing the model and gradient recursions respectively. The gain sequence is denoted $\gamma(t)$, and $e(t)$ is a sequence of random vectors.

The following regularity conditions on (5.5) are introduced in [53], section 4.2. $D_R$ is a subset of the $x_g$-space, where the regularity conditions are assumed to hold.

C1: $\|g(x_g, \varphi, e)\| < C \quad \forall \varphi, e \quad \forall x_g \in D_R$.

C2: $Q(t, x_g, \varphi_g)$ is continuously differentiable with respect to $x_g$ and $\varphi_g$, and the derivatives are bounded in $t$ for $x_g \in D_R$.

C3: $g(t; x_g, \varphi_g, e)$ is continuously differentiable with respect to $x_g$ for $x_g \in D_R$.

C4: Define $\bar{\varphi}_g(t, \bar{x}_g)$ as $\bar{\varphi}_g(t, \bar{x}_g) = g(t; \bar{x}_g, \varphi_g(t - 1, \bar{x}_g), e(t))$, $\bar{\varphi}_g(0, \bar{x}_g) = 0$ and assume that $g(\cdot)$ has the property

$$\|\bar{\varphi}_g(t, \bar{x}_g) - \varphi_g(t)\| < C \max_{n \leq k \leq t} \|\bar{x}_g - x_g(k)\|$$

if $\bar{\varphi}_g(n, \bar{x}_g) = \varphi_g(n)$.

C5: Let $\bar{\varphi}_{g,i}(t, \bar{x}_g)$ be solutions of (5.3) with $\bar{\varphi}_{g,i}(s, \bar{x}_g) = \varphi^0_{g,i}$, $i = 1, 2$. Then define $D_S$ as the set of all $\bar{x}_g$ for which holds $\|\bar{\varphi}_{g,1}(t, \bar{x}_g) - \bar{\varphi}_{g,2}(t, \bar{x}_g)\| < C(\varphi^0_{g,1}, \varphi^0_{g,2}) \lambda^{t-s}(\bar{x}_g)$ where $t > s$ and $\lambda(\bar{x}_g) < 1$. (Region of exponential stability of (5.5))

C6: $\lim_{t \to \infty} E Q(t, \bar{x}_g, \bar{\varphi}_g(t, \bar{x}_g)) = f(\bar{x}_g), \bar{x}_g \in D_R$, with expectation over $e(\cdot)$, exists.

C7: $e(\cdot)$ is a sequence of independent random variables.

C8: $\sum_{t=1}^{\infty} \gamma(t) = \infty$.

C9: $\sum_{t=1}^{\infty} \gamma^p(t) < \infty$ for some $p$. 
5.4 Conditions on the Algorithm and the Data

When analyzing the algorithm (5.4), the following regularity conditions on the model, the algorithm, and the data are used.

M1: $D_M$ is a compact subset of $\mathbb{R}^{d+d^2+p^2}$, such that the extended parameter vector $\theta^* = (\theta^T (\text{col}(R))^T (\text{col}(\Lambda))^T)^T \in D_M$ implies that i) the combined dynamics of (5.4) is exponentially stable and bounded, ii) 

C10: $\gamma(\cdot)$ is a decreasing sequence.

C11: $\lim_{t \to \infty} \sup(\frac{1}{\gamma(t)} - \frac{1}{\gamma(t-1)}) < \infty$.

It is then proven in [53] that, provided that C1-C11 hold, the theorems and averaging analysis tools of [53] (and [54]) are applicable to (5.5). The details are clarified in Section 5.5.

In Fig. 5.1 a schematic picture of the analysis in [54], [53] is shown to the right. To the left is the parallel of each step for the RPEM algorithm of this analysis. The RPEM algorithm (5.4) can be written as (5.5), and the regularity conditions C1-C11 (from [53]) are fulfilled provided that conditions M1, M2, G1, A1, A2, S1, and S2 of Section 5.4 hold for (5.4). Lemma 3 and 4 are identical to Corollary 1 and Theorem 2 of [54], which enables the convergence analysis tools in [54] to be applied.

Figure 5.1: A schematic picture of the convergence analysis procedure of the RPEM (left) as compared to that of Ljung (1975) (right).
The reason for the introduction of a condition on exponential stability is that exponentially stable and bounded models and gradients for a fixed value of \( \theta^* \) are required to verify \( C7 \) stringently; at the same time it would be too restricted to require that the input is white noise. In this respect, the present analysis extends the one in [17], where the state extension was only outlined. Further, the input signal derivatives are not allowed. M2: 

\[
R(t) \geq \delta_R I, \ \forall t, \text{ some } \delta_R > 0, \ iii) \ \Lambda(t) \geq \delta_\Lambda I, \ \forall t, \text{ some } \delta_\Lambda > 0. \text{ Furthermore (5.3) guarantees } \theta^* \in D_M \ \forall t.
\]

M2: \( u(t) = (\mathbf{u}_1(t) \ldots \mathbf{u}_k(t))^T \), i.e. input signal derivatives are not allowed. Further, \( u(t) = C_u \mathbf{x}_u(t) \), where the states \( \mathbf{x}_u \) are generated from \( \bar{u}(t) \) by a continuously differentiable, bounded, exponentially stable difference equation. \( \bar{u}(t) \) is a sequence of independent identically distributed (i.i.d.), bounded random vectors.

G1: \( \lim_{t \to \infty} \mu(t) = \mu > 0. \)

A1: The data sequence \( \{ z(t) \} = (y_m^T(t) u^T(t))^T \) is strictly stationary and such that, w.p.1, \( \|z(t)\| \leq C < \infty, \ \forall t. \)

A2: The following limits exist for fixed \( \theta^* \in D_M \)

\[
\begin{align*}
\lim_{t \to \infty} E\psi(t, \theta)\Lambda^{-1} \varepsilon(t, \theta) &= f(\theta, \Lambda), \\
\lim_{t \to \infty} E\psi(t, \theta)\Lambda^{-1} \psi^T(t, \theta) &= G(\theta, \Lambda), \\
\lim_{t \to \infty} E\varepsilon(t, \theta)\varepsilon^T(t, \theta) &= H(\theta).
\end{align*}
\]

S1: For each \( t, s, t \geq s \), there exists a random vector \( z^s(t) \) that belongs to the \( \sigma \)-algebra generated by \( z^t \) but is independent of \( z^s \) (for \( s = t \) take \( z^t(0) = 0 \)), such that \( E\|z(t) - z^0(0)\|^4 < C\lambda^{t-s}, \ C < \infty, \ |\lambda| < 1. \)

S2: The system that generates the data can be described by an ODE with bounded dynamics (i.e. C1 holds for the data generation). Further, \( y_m(t) = C_{y_m} x_{y_m}(t) + C_w x_w(t) \), where the states \( x_{y_m} \) are generated from \( u(t) \) by a continuously differentiable, bounded, exponentially stable difference equation, and where the states \( x_w \) are generated from \( \bar{w}(t) \) by bounded linear asymptotically stable filtering. The signal \( \bar{w}(t) \) is a sequence of bounded i.i.d. random vectors, independent of \( \bar{u}(t) \).

The imposed conditions can be explained as follows. M1 defines the set of exponentially stable and bounded models and gradients for a fixed value \( \theta^* \in D_M \). The reason for the introduction of a condition on exponential stability is that for nonlinear systems local exponential stability (in \( D_M \)) only results in local boundedness around equilibrium points, where the region of boundedness may be smaller than \( D_M \) ([44], Theorem 6.1). It is only for globally exponentially stable models that global boundedness is guaranteed ([44], Theorem 6.1). However, it is very likely that there exist interesting non-globally exponentially stable systems that can be described by the polynomial model of (5.4). Hence, a boundedness assumption is introduced in the present analysis. M2 is introduced in order to obtain a bounded combined system that is driven by i.i.d random vectors. The introduction of this model for the input signal generation is required to verify C7 stringently; at the same time it would be too restricted to require that the input is white noise. In this respect, the present analysis extends the one in [17], where the state extension was only outlined. Further,
the condition G1 is standard. The conditions on the data express boundedness (A1), exponential stability (S1), and a basic assumption on the existence of the average updating directions (A2). This avoids lengthy discussions on ergodicity that would otherwise be needed. Since the vector $\varphi_g(t)$ of (5.5) contain the data, the last condition (S2) is required to ensure that C1 holds. The extension of S2 as compared to [17] is motivated by the same reason as stated when M2 was discussed above. Hence, a model where $u$, $y_m$, and $w$ are generated by quite general filters is assumed in the analysis.

5.5 Results

5.5.1 Outline of Proof

In order to prove the main results, (5.4) is first written in the form of (5.5) using M1, M2, and S2 (Lemma 1). The regularity conditions C1-C11 are then verified for (5.6) of Lemma 1, using M1, M2, G1, A1, A2, S1, and S2. This proves Lemma 2, i.e. Theorem 8 of [53]. Lemma 2 shows that Theorem 2 and Corollary 1 of [54] can be used for analysis of (5.4). The conditions of these theorems are then verified, which leads to the validity of Lemma 3 and 4. Further analysis using Lemma 3 and 4 establishes the main result of Theorem 1.

5.5.2 Establishing the Tools for Analysis

The result of the first part of the analysis is given by

**Lemma 1.** The algorithm (5.4) can be transformed to the form (5.5), using the definitions (A.1)-(A.11) together with

\[
x_g = \theta^* \quad \varphi_g = \xi \quad \gamma(t) = \frac{1}{t} \quad e(t) = \bar{z}(t) = (\bar{u}^T(t) \bar{w}^T(t))^T
\]

\[
Q(t, x_g, \varphi_g) = \begin{pmatrix}
\mu(t)R^{-1}\zeta_\psi(\xi)\Lambda^{-1}\zeta_\varepsilon(\xi)\text{gate}(\Gamma_\theta(\theta^*)) \\
\mu(t)\text{col}(\zeta_\psi(\xi)\Lambda^{-1}\varepsilon^T_\psi(\xi) - R)\text{gate}(\Gamma_R(\theta^*)) \\
\mu(t)\text{col}(\zeta_\varepsilon(\xi)\varepsilon^T_\varepsilon(\xi) - \Lambda)\text{gate}(\Gamma_\Lambda(\theta^*))
\end{pmatrix},
\]

\[
g(t, x_g, \varphi_g, e) = \begin{pmatrix}
\eta_x(\theta^*, \xi(t - 1), \bar{z}(t)) \\
\eta_{\Delta \theta}(\theta^*, \xi(t - 1), \bar{z}(t)) \\
\kappa_{\bar{y}}(\xi(t - 1), \bar{z}(t)) \\
\kappa_u(\xi(t - 1), \bar{z}(t)) \\
\kappa_w(\xi(t - 1), \bar{z}(t)) \\
\bar{z}(t)
\end{pmatrix}
\]  

(5.6)

where gate($\cdot$) is the gate function defined by

\[
gate(\Gamma) = \begin{cases}
1 & \Gamma \in I_o = (\Gamma_-, \Gamma_+) \\
0 & \Gamma \notin I_o = (\Gamma_-, \Gamma_+)
\end{cases}
\]  

(5.7)

$\Gamma(\cdot)$ maps arguments onto $I_o$ when the arguments $\in D_M$ and outside $I_o$ when the arguments $\notin D_M$. Typically $\Gamma_\theta$ is selected as the largest absolute value of
the eigenvalues of the linearized system, and \( I_o = (0, 1 - \delta) \). The gate function shows that the projection can be incorporated in (5.5), cf. the discussion of [92].

**Proof:** See Appendix A.1.

**Lemma 2.** Consider the algorithm (5.4) and assume that \( M1, M2, G1, A1, A2, S1 \) and \( S2 \) hold. Then Corollary 1 and Theorem 2 of [54] hold.

**Proof:** See Appendix A.2

**Remark:** It is known from Lemma 1 that the algorithm (5.4) can be written as (5.5), which is treated in [53] under the conditions C1-C11. It is now shown in Appendix A.2 that when \( M1, M2, G1, A1, A2, S1 \) and \( S2 \) hold the conditions C1-C11 are also fulfilled. This means that according to Theorem 8 in [53], Lemma 1 and Theorems 1-6 in [53] are applicable to (5.4). Theorems 1 and 2 of [54] are exactly the same as Theorems 1 and 5 of [53], respectively, and the conclusion is that these theorems can therefore be used when analyzing (5.4). Also, Corollary 1 of [54] follows from Theorem 1 of [54], and can thus also be used for the continued analysis of (5.4). The above analysis prove that Lemma 3 and 4 below are valid for the continued analysis of (5.4). Lemma 3 and 4 are the main tools of analysis of this chapter.

**Lemma 3.** (Corollary 1 of [54]) Consider the algorithm (5.4) and assume \( M1, M2, G1, A1, A2, S1 \) and \( S2 \) hold. Also assume that there exists a twice differentiable positive function \( U(\theta, R, \Lambda) \) such that

\[
\frac{d}{d\tau} V(\theta_D(\tau), R_D(\tau), \Lambda_D(\tau)) \leq 0
\]

for \((\theta_D, R_D, \Lambda_D) \in D_M \setminus \partial D_M\) (5.8)

when evaluated along solutions of the differential equations

\[
\frac{d}{d\tau} \theta_D(\tau) = \mu R_D^{-1}(\tau)f(\theta_D(\tau), \Lambda_D(\tau))
\]
\[
\frac{d}{d\tau} R_D(\tau) = \mu(G(\theta_D(\tau), \Lambda_D(\tau)) - R_D(\tau)), \quad (5.9)
\]
\[
\frac{d}{d\tau} \Lambda_D(\tau) = \mu(H(\theta_D(\tau)) - \Lambda_D(\tau))
\]

where \( f(\theta, \Lambda), G(\theta, \Lambda), H(\theta) \) are defined by A2. Let

\[
D_C = \left\{ (\theta_D, R_D, \Lambda_D) \mid (\theta_D, R_D, \Lambda_D) \in D_M \setminus \partial D_M, \right. \]
\[
\left. \frac{d}{d\theta} V(\theta_D(\tau), R_D(\tau), \Lambda_D(\tau)) = 0 \right\}. \quad (5.10)
\]

Then either

\[
(\hat{\theta}(t), R(t), \Lambda(t)) \to D_C \text{ w.p.1 as } t \to \infty, \quad (5.11)
\]

or \((\hat{\theta}(t), R(t), \Lambda(t))\) converges to the boundary \( \partial D_M \).
Proof: Follows from Lemma 1 and 2, and the analysis of Appendix A.3

Remark: The subscript \( \rho \), e.g. in \( \theta_D(\tau) \), denotes state variables of the associated ODE, whereas \( \tau \) denotes a transformed time, see [53; 54; 58] for details.

Remark: Lemma 3 corresponds to Corollary 1 of Theorem 1 in [54], with the exception that here the boundedness condition is not mentioned as a condition. The reason is that it is shown in Appendix A.3 that under the conditions above, the boundedness condition automatically holds.

**Lemma 4. (Theorem 2 of [54])** Consider the algorithm (5.4) subject to M1, M2, G1, A1, A2, S1, and S2. Suppose that \( \tilde{\varphi}_g(= (\tilde{\theta}, \tilde{\Lambda})) \in D_M \setminus \partial D_M \) and the gradient \( \nabla f(\tilde{\theta}, \tilde{\Lambda}) \) has all eigenvalues in the left half-plane \( \text{Re}(z) \leq 0 \).

Proof: Follows from Lemma 1 and 2.

### 5.5.3 The Main Result

The next step is to prove global convergence to a minimum point of the criterion using Lemma 3. First note that the criterion function is (cf. [58]).

\[
V(\theta, \Lambda) = \frac{1}{2} E \varepsilon^T(\theta) \Lambda^{-1} \varepsilon(t, \theta) + \frac{1}{2} \log \det \Lambda. \tag{5.13}
\]

Note that \( f(\theta, \Lambda) = -\left( \frac{\partial}{\partial \sigma} V(\theta, \Lambda) \right)^T \), and that \( \frac{\partial}{\partial \sigma} \log \det \Lambda(\beta) = \text{tr}(\Lambda^{-1} \frac{\partial}{\partial \sigma} \Lambda(\beta)). \)

By parallel arguments as in [58], p. 187, it follows that

\[
\frac{d}{d\tau} V(\theta_D(\tau), \Lambda_D(\tau)) =
- \mu f^T(\theta_D(\tau), \Lambda_D(\tau)) R_D^{-1}(\tau) f(\theta_D(\tau), \Lambda_D(\tau))
- \frac{1}{2} \text{tr} \left( \Lambda_D^{-\frac{1}{2}}(\tau) (\mu(H(\theta_D(\tau))) - \Lambda_D(\tau)) \Lambda_D^{-\frac{1}{2}}(\tau) \right) \mu(H(\theta_D(\tau))) - \Lambda_D(\tau)) \Lambda_D^{-\frac{1}{2}}(\tau), \tag{5.14}
\]

where \( \Lambda_D^{-\frac{1}{2}} \) is a symmetric square root of \( \Lambda_D^{-1} \). Since \( \Lambda_D \), and \( R_D \) are positive definite (from M1)

\[
\frac{\partial}{\partial \tau} V(\theta_D(\tau), \Lambda_D(\tau)) \leq 0 \tag{5.15}
\]
where the equality holds only for \( \theta, \Lambda \) such that
\[
f(\theta, \Lambda) = -\left( \frac{\partial}{\partial \theta} V(\theta, \Lambda) \right)^T = 0, \tag{5.16}
\]
\[
H(\theta) - \Lambda = 2 \frac{\partial}{\partial \Lambda^{-1}} V(\theta, \Lambda) = 0, \tag{5.17}
\]
i.e., for stationary points of \( V(\theta, \Lambda) \). Hence, according to Lemma 3 global convergence to a set described by (5.16)-(5.17) holds for the algorithm (5.4).

It remains to investigate local convergence to stationary points using Lemma 4. First the following conditions are introduced

M3: The observations \( z(t) \) are generated by a system that can be described by (5.1), (5.2), i.e. there exists a true extended parameter vector \( \tilde{\theta}_0^* = (\theta_0 \ col R_0 \ col \Lambda_0)^T \in D_M \setminus \partial D_M \), such that the estimation error \( \varepsilon(t, \theta_0) \) is white noise, independent of the inputs, with covariance matrix \( \Lambda \).

A3: \( \frac{\partial}{\partial \theta} f(\theta, \Lambda) |_{\theta^*, \theta_0^*} = E\psi(t, \theta_0)\Lambda_0^{-1}\psi^T(t, \theta_0) = G(\theta_0, \Lambda_0) = R_0 > 0 \)

Remark: A3 states that the Hessian is positive definite in the true parameter vector. Fulfillment on this condition is likely to require conditions on persistent excitation and identifiability, generalized from the linear case. This is a subject for future research.

From A2 and M3 follows that
\[
(\theta^T \ col(\tilde{R}^T) \ col(\tilde{\Lambda}^T)) = (\theta_0^T \ col(T) \ col(T)(\Lambda_0))
\]
is a stationary point as \( f(\theta_0, \Lambda_0) = 0, \ G(\theta_0, \Lambda_0) = R_0 \), and \( H(\theta_0) = \Lambda_0 \).

Provided that A3 holds, the expression
\[
\frac{1}{\mu} \nabla \bar{f} = \begin{pmatrix}
\frac{\partial}{\partial \theta} R^{-1} f(\theta, \Lambda) & \frac{\partial}{\partial \theta} R^{-1} f(\theta, \Lambda) & \frac{\partial}{\partial \theta} R^{-1} f(\theta, \Lambda) \\
\frac{\partial}{\partial \theta} (G(\theta, \Lambda) - R) & \frac{\partial}{\partial \theta} (G(\theta, \Lambda) - R) & \frac{\partial}{\partial \theta} (G(\theta, \Lambda) - R) \\
\frac{\partial}{\partial \theta} (H(\theta) - \Lambda) & \frac{\partial}{\partial \theta} (H(\theta) - \Lambda) & \frac{\partial}{\partial \theta} (H(\theta) - \Lambda)
\end{pmatrix}
\]
\[
= - \begin{pmatrix}
I & 0 & 0 \\
2E \frac{\partial}{\partial \theta} \psi(t, \theta)\Lambda^{-1}\psi^T(t, \theta) & I & 0 \\
0 & E\psi(t, \theta)\Lambda^{-2}\psi^T(t, \theta) & I
\end{pmatrix}
\tag{5.18}
\]
then holds. Most of the calculations of (5.18) are straightforward; e.g. the identity matrices of the diagonal elements follow from differentiation of \( \bar{f} \) and the use of the expressions in A2. The corner elements of the anti-diagonal are zero, since \( \varepsilon \) and \( \psi \) are independent in the stationary point (follows from M3 and the fact that (5.4) is of output error type, i.e. generated from the input signal and parameters). Further, the middle element in the top row becomes zero in the stationary point as \( f(\theta_0, \Lambda_0) = 0 \), and the middle element in the bottom row is zero since the differentiated expression does not contain \( R \). In [79] it is shown that the determinant of a block matrix
\[
P = \begin{pmatrix}
A & B \\
C & D
\end{pmatrix},
\tag{5.19}
\]
where $A$ and $D$ are square matrices, can be written as $\det P = \det A \det(D - CA^{-1}B)$. This implies that

$$\det \nabla \tilde{f} = (-\mu)^{d+d^2+r^2} \det(I) \neq 0. \quad (5.20)$$

Consequently, from (5.18), $\nabla \tilde{f}$ is negative definite, so all eigenvalues are in the left half plane. Thus the stationary point $(\theta_0, R_0, \Lambda_0)$ is stable, and the true parameter set is a possible convergence point for (5.4) as stated by Lemma 4. This proves the main result.

**Theorem 1.** Consider the algorithm (5.4) and assume that $M1, M2, M3, G1$, $A1, A2, A3, S1$ and $S2$ hold. Then $(\hat{\theta}(t) \col \Lambda(t))^T$ converges w.p.1 either to the set of stable stationary points of $V(\theta, \Lambda)$, or to $\partial D_M$, as $t \to \infty$.

**Remark:** The algorithm is formulated for the model structure (5.1)-(5.2), where $\varphi^T(x, u)\theta$ is a polynomial in $x$ and $u$. However, the general formulation of the conditions $M1, M2, M3, G1, A1, A2, A3, S1$ and $S2$, suggest that the analysis is valid for a much wider range of nonlinearities.

### 5.6 Numerical Example

To illustrate the results above a numerical example described in [4] is used. The example corresponds to a temperature servo for heating a rod, operated in closed loop. Heat is transported to and from the rod by a Peltier effect device, which has nonlinear dynamics due to energy loss caused by resistance. The system can be described by the state space representation

$$\dot{x} = \begin{pmatrix} x_2 \\ -x_2 + u - x_1 + (u - x_1)^2 \end{pmatrix}, \quad y = x_1 \quad (5.21)$$

which is of the same form as (5.1), (5.2). The simulations were performed using the software implementation in [94] of the algorithm. Data was generated using a PRBS-like input signal with uniformly distributed amplitudes, and the sampling time $0.05s$. Studying the model set containing the true system it becomes clear that most of the 18 parameters should be zero in the global minimum. However, with that many parameters there also appears to be several local minima that the algorithm can converge to. As the objective of this numerical example is to illustrate that the “true” parameter set is a possible convergence point for the algorithm, the number of parameters in $\theta$ are reduced to containing a bias parameter and the parameters included in the system dynamics (5.21). Hence $\varphi(x, u)$ and the “true” parameter vector $\theta_0$ become

$$\varphi(x, u) = (1 \quad u \quad u^2 \quad x_2 \quad x_1 \quad x_1 u \quad x_1^2)^T \quad (5.22)$$

$$\theta_0 = (0 \quad 1 \quad 1 \quad -1 \quad -1 \quad -2 \quad 1)^T. \quad (5.23)$$

The RPEM was initialized with $\Lambda(0) = 0.1$, and $R(0) = 10I$. $x_1(0)$ was chosen as the sum of the mean and the standard deviation of $y$, $x_2(0) = 0,$
\( \frac{\alpha}{\delta_0} \) = 0. Further, the scale factor \( \alpha = 2 \), and the stability limit of the projection algorithm was \( \hat{\delta} = 0.025 \). The initial parameter vector was chosen as \( \hat{\theta}(0) = (0 \ 0 \ 0 \ -0.9 \ -0.9 \ 0 \ 0)^T \). At the end of the run the scaled and rescaled parameters (corresponding to \( \theta_0 \)) were

\[
\hat{\theta}(10000T_S) = (0.0005 \ 0.2666 \ 0.2429 \ -0.5140 \\
-0.2656 \ -0.5037 \ 0.2590)^T \tag{5.24}
\]

\[
\hat{\theta}_{rescaled}(10000T_S) = (0.0022 \ 1.0666 \ 0.9718 \ -1.0280 \\
-1.0623 \ -2.0149 \ 1.0360)^T \approx \theta_0, \tag{5.25}
\]

see (3.21) [97] and [96] for the transformation between scaled and original parameters. The parameter convergence is illustrated in Fig. 5.2.

### 5.7 Summary

In this chapter the convergence properties of the RPEM in Chapter 3 have been analyzed. A number of conditions were introduced, and it was proven that these conditions imply convergence to a minimum of the RPEM criterion. The conditions allow for fairly general generation of inputs, outputs and disturbances, using the stochastic framework of [53; 54]. The parameters that the algorithm converges to may correspond to the global or a local minimum of the criterion function. In particular, the algorithm converges locally to the true parameter vector, provided that the corresponding Hessian is non-singular.
These results provide an indication of under what conditions the algorithm can be expected to converge. As for any nonlinear identification problem with a non-convex criterion function, it is very difficult to ensure that the algorithm converges to the global minimum of the criterion function. In order to improve the chances of convergence to the true parameter vector, the initial parameters of the algorithm should be chosen close to the global minimum. This supports the conclusion in Chapter 4, concerning the need for a method for finding initial parameters for the RPEM. This issue is addressed next.
Chapter 6

An Initialization Algorithm

As indicated by the examples in Chapter 4 and the convergence analysis in Chapter 5, the use of the RPEM in Chapter 3 would be simplified if there was a method for finding suitable initial parameters for the algorithm. This is not a new problem; many other general nonlinear identification algorithms risk convergence to suboptimal minima of the criterion function depending on the choice of initial parameters, cf. [55; 58]. Least squares and similar techniques have a distinct advantage as compared to nonlinear search methods that are often applied in output error and grey-box identification. The advantage manifests itself in that least squares methods avoid the risk of convergence to sub-optimal minima of the criterion function [58].

A main objective of this chapter is to obtain an algorithm that is suitable for initialization of the RPEM of Chapter 3, which sometimes may converge to false suboptimal minima of the criterion function. Contrary to the previously developed RPEM [15–17; 96; 97; 99], where the regressors are modeled by output error techniques, the regressors of the proposed identification algorithm are obtained by numerical differentiation of the measurements. The new algorithm allows for standard numerical differentiation using the Euler method, as well as for more elaborate differentiation schemes [68]. Due to the above facts, the new method has similar convergence properties as the class of equation error methods and does not converge to sub-optimal minimum points. The price paid is that the estimates may become biased. However, as stated above, the proposed algorithm can with advantage be used for initialization of the RPEM, thereby reducing the risk that the RPEM converges to a false optimum. The obtained advantages are further quantified in Section 6.2.2 below. An alternative to the proposed algorithm could be to apply a linear least squares method to obtain an initializing linear model. This could be done using a discrete time or a continuous time model of the system. When a discrete time model is used, the transformation to continuous time may encounter problems though, due to non-physical zeros [6]. In case a continuous time linear model would be used, the proposed nonlinear algorithm provides a more general approach, since it
estimates continuous time parameters. A final important advantage is that the proposed method can exploit scaling of the sampling period. This scaling method can improve the conditioning of the identification problem with several orders of magnitude as shown in [96].

6.1 A Kalman Filter Based Identification Algorithm

6.1.1 Generation of Regressors

The Kalman filter based algorithm described in this section is based on the same model structure as the RPEM (3.17). The model structure is

\[
\begin{bmatrix}
  x_1^{(1)} \\
  \vdots \\
  x_n^{(1)} \\
  x_{n-1}^{(1)} \\
  x_n^{(1)}
\end{bmatrix}
= \begin{bmatrix}
  x_2 \\
  \vdots \\
  x_n \\
  f(x, u, \theta)
\end{bmatrix}
\]

(6.1)

\[y = (1 \ 0 \ldots \ 0)x,\]

(6.2)

where

\[f(x, u, \theta) = \varphi^T(x, u)\theta\]

(6.3)

is a polynomial as before. This is natural since the objective of the Kalman filter based algorithm is to provide initial parameters for the RPEM. The model structure is, however, used slightly differently in the two algorithms, and as a result the regressor vectors will not be identical.

The regressors are, as can be seen in (3.7), generated from the states, \(x\), and the inputs, \(u\). Clearly the inputs are measured, while the states need to be estimated. A first method, that result in the RPEM [97] is to update the whole state vector by numerical integration as in (3.15). A second method, that results in the new algorithm proposed in this paper, is to estimate the states by numerical differentiation of the measured output. The reasoning behind the latter method is described next.

Each state of (3.15) corresponds to the derivative of the previous state. Hence, a simple numerical differentiation of the measured output signal should render an approximation of \(x_2\). Similarly, by numerical differentiation of the second state, an approximation of the third state would be obtained, and so on. Hence, if \(\xi_1(t)\) is set equal to the measured output \(y_m(t)\) the approximated states \(\xi(t)\) can be obtained by iteration

\[
\begin{align*}
\xi_1(t) &= y_m(t) \\
\xi_k(t) &= \frac{\xi_{(k-1)}(t) - \xi_{(k-1)}(t - T_s)}{aT_s}, \quad k = 2, \ldots, n \\
\xi(t) &= (\xi_1(t) \ldots \xi_n(t))^T.
\end{align*}
\]

(6.4)

Note the inclusion of scaling in the differentiation. If required, the Euler differentiation method applied in (6.4) can be exchanged for a more advanced
differentiating filter. It is well known that differentiation is associated with sensitivity to noise. In order to reduce the noise transmission a low-pass filtering of the states may be feasible. A general description of a $n^{th}$ order differentiation scheme is given by

$$
\xi_1(t) = y_m(t)
$$

$$
\xi_k(t) = H^d(q^{-1}, \alpha) H^L_k(q^{-1}, \alpha) \xi_{k-1}(t), \quad k = 2, ..., n. \quad (6.5)
$$

where $H^L_k(q^{-1}, \alpha), \ k = 2, ..., n$ are digital filters, typically of low-pass character (see [68] for further details), and $H^d(q^{-1}, \alpha)$ is the differentiating filter, which in its simplest form is given by

$$
H^d(q^{-1}, \alpha) = \frac{1 - q^{-1}}{\alpha T_s}. \quad (6.6)
$$

Note that the filters depend on the order of the system.

The main motivation for introducing the numerical differentiation lies in the fact that $x(t, \theta)$ essentially is obtained by numerical integration originating in $f(x, u, \theta)$, whereas $\xi(t)$ consists of derivatives of the measured output signal. This implies that by exchanging $x(t, \theta)$ for $\xi(t)$ in the nonlinear function $f(x, u, \theta)$ the regressors cease to depend on the estimated parameters $\theta$. Consequently, the optimization problem becomes linear in the parameters, with a convex criterion function. This is important since it addresses the main problem of the RPEM method considered here - the RPEM criterion function which may have multiple sub-optimal minima. Depending on the choice of initial values the RPEM may therefore converge to a local minimum rather than the global minimum. It should be noted that by exchanging $x(t, \theta)$ for $\xi(t)$ in $f(\cdot, \cdot, \cdot)$ the output error model, as used in the RPEM, can be replaced by an equation error model or a Kalman filter. The disadvantage of this change is an increased sensitivity to unmodeled dynamics [87]. It can also be assumed that differentiation of the measurement noise will cause problems for low signal to noise ratios (SNR). However, as stated above, the main purpose is to use the new method for initialization of the RPEM.

### 6.1.2 Algorithm

As stated above the algorithm used for initialization of the RPEM is based on the Kalman filter, see e.g. [79]. The idea is to use the approximated states, $\xi(t)$, obtained by differentiation of the measured output to determine a linear system matrix $F$. To achieve this the state vector is first extended to also include the parameter vector

$$
\bar{x}(t) = (x_1(t, \theta) \ldots x_n(t, \theta) \ \theta^T(t))^T. \quad (6.7)
$$
The right hand side of (3.15) can then be approximated as \( F(t, \xi, u)\hat{x}(t) \). The system matrix \( F(t, \xi, u) \) and the output matrix \( H \) (cf. (3.16)) are given by

\[
F(t, \xi, u) = \begin{pmatrix}
1 & \alpha T_s & \cdots & 0 & 0 \\
\vdots & \ddots & \ddots & \vdots & \vdots \\
\vdots & \vdots & 1 & \alpha T_s & 0 \\
0 & \cdots & 0 & 1 & \alpha T_s \varphi(\xi(t), u(t)) \\
0 & \cdots & 0 & I_{n_s} & \end{pmatrix}
\] (6.8)

\[
H = (1 \ 0 \ldots 0).
\] (6.9)

The parameter \( \alpha \) is the scale factor applied to the sampling period, cf. Section 3.4. Writing out the Kalman filter based method as an algorithm, see [78], results in

Initiate:
\[
\hat{x}(0| - T_s), P(0| - T_s)
\]

Iterate:
\[
\begin{align*}
K(t) &= P(t|t - T_s)H^T(HP(t|t - T_s)H^T + R_2)^{-1} \\
\hat{x}(t|t) &= \hat{x}(t|t - T_s) + K(t)(y_m(t) - H\hat{x}(t|t - T_s)) \\
P(t|t) &= P(t|t - T_s) - K(t)HP(t|t - T_s) \\
\hat{x}(t + T_s|t) &= F(t, \xi(t), u(t))\hat{x}(t|t) \\
P(t + T_s) &= F(t, \xi(t), u(t))P(t|t)F^T(t, \xi(t), u(t)) + R_1.
\end{align*}
\] (6.10)

Here \( R_1 \) and \( R_2 \) are the covariance matrices of the process noise and measurement noise, respectively. In the following \( R_1 \) will be on the form

\[
R_1 = \begin{pmatrix}
R_{1,x} & 0 \\
0 & R_{1,\theta}
\end{pmatrix}
\] (6.11)

Note that in (6.8) the estimated parameters only occur in \( \hat{x}(t) \). Note also that it is only in the right hand-side of the ODE that the approach \( x(t, \theta) \approx \xi(t) \) is used. Note also that the results on scaling of [96] apply to (6.10) as well.

### 6.2 Experiments

#### 6.2.1 Accuracy Comparison

In order to evaluate and compare the accuracy of the algorithms (3.17) and (6.10) data was generated by simulation of a system corresponding to the model structure (6.1), with measurement noise added to the output (6.2). The studied second order system

\[
\begin{pmatrix}
x_1^{(1)} \\
x_2^{(1)}
\end{pmatrix} = \begin{pmatrix}
x_2(2 + u) - u \\
x_1 - x_2
\end{pmatrix}
\]

\[ y = x_2 + \epsilon 
\] (6.12)
has previously been described in [96], and can be rewritten on the form of (6.1)-(6.2). To generate the data, an Euler integration method was used for updating the states. To ensure that the numerical method used for data generation was not part of the model set, the data was generated with a sampling period of $1.0 \cdot 10^{-4}$ s. The identification was then performed using subsets of the total data set with sampling periods between $5.0 \cdot 10^{-3}$ s and $5.0 \cdot 10^{-1}$ s, thereby ensuring that discretization errors were generated. The reason was to make sure the bias that can occur in live data sets due to discretization effects become visible also in the numerical example. For a system that is part of the model set this would not be the case.

The model orders were chosen as $I_u = I_{x_1} = I_{x_2} = 1$ which is sufficient to express (6.12) on the form (6.1)-(6.2). The Kalman filter based algorithm was initialized with $\hat{\theta}(0) = 0$. $P(0) - T_s$, $R_1$ and $R_2$ were tuned in the various $SNR$ and $T_s$ cases, while the scale factor $\alpha = 1$ was kept constant. The RPEM algorithm was initialized with $\Lambda(0) = 0.2I$, $R(0) = 10^4$, $\mu_0 = 0.9995$, $\mu_1 = 300$, $\mu(0) = 5$, and $\delta = 0.001$.

To measure the performance of the algorithms, the following error measures were used

$$V = \frac{||\hat{\theta}(N) - \theta_0||_2}{||\theta_0||_2},$$

(6.13)

$$MSE = \frac{1}{N} \sum_{t=1}^{N} (y_{m,0}(t) - y(t))^2$$

(6.14)

where $\hat{\theta}(N)$ denotes the estimated parameter vector at the end of the run, and $\theta_0$ is the true continuous time parameter vector from which the data has been generated. $y_{m,0}(t)$ is the measured output signal for the noise free case, and $y(t)$ is the simulated output obtained with the model corresponding to $\hat{\theta}(N)$. The two error measures illustrate both the the misalignment in the estimated parameters and the accuracy of the simulated model output. The purpose of the comparison to the noise free output signal is to illustrate the differences of the methods without obscuring modeling errors with pure noise. The parameter convergence of the Kalman filter based algorithm for the noise free data case can be seen in Fig. 6.1

**Noise Sensitivity**

Identification experiments with the Kalman filter based and RPEM algorithms show that both algorithms perform better for higher SNR (see Fig. 6.2 and 6.3). This seems reasonable, as does the fact that the curves level out when the SNR is no longer the limiting factor for algorithm performance. The break point for noise sensitivity is approximately 10 dB higher for the Kalman filter based method than for the RPEM. This can be explained by the state estimation methods, where differentiation of the measured output (Kalman filter based) will produce an algorithm more sensitive to measurement noise than when numerical integration is used to generate the regressor vectors (RPEM).
Low-pass Filtering

The standard Euler differentiation scheme was also compared to a low-pass differentiation scheme (6.5). In this example the filter $H_L^P(q^{-1}, \alpha)$ was selected as a second order Butterworth low-pass filter with normalized (w.r.t. $T_s$) bandwidth, and $H^d(q^{-1}, \alpha)$ was given by (6.6). As can be seen in Fig. 6.2 and Fig. 6.3, low-pass filtering may extend the operating range 1-3 dB towards lower SNRs. However, the performance remains worse than the performance of the RPEM. Furthermore, there is a performance penalty as compared to the Euler differentiation scheme for high SNRs. The explanation is that the low-pass filtering removes useful information from the signals in the high SNR case.

In order to illustrate the properties of (6.5) further, the performance of the Kalman filter based algorithm is displayed as a function of the normalized bandwidth in Fig. 6.4 and 6.5. As expected, a distinct optimum appears for low SNRs, whereas the experiments indicate that low-pass filtering should be avoided for higher SNRs. In the rest of the experimental investigation of the paper, the Euler differentiation method is applied.

Effects of the Sampling Period

As for the variation of $T_s$ the effect on the parameter estimate as well as the prediction error shows a minimum for the Kalman filter based algorithm, see Fig. 6.6 and 6.7. Variation of the sampling period illustrates the trade-off
Figure 6.2: The effect of the signal to noise ratio (SNR) on parameter estimation for the Kalman filter based algorithm (solid), and the RPEM (dashed). The dash-dotted line corresponds to the Kalman filter based algorithm with low-pass filtering (normalized bandwidth $0.6\pi$ rad/s). The variables $T_s$ and $\alpha$ were kept fixed at 0.1 s and 1, respectively.

Figure 6.3: The effect of the signal to noise ratio (SNR) on the mean square of the prediction error for the Kalman filter based algorithm (solid) and the RPEM (dashed). The dash-dotted line corresponds to the Kalman filter based algorithm with low-pass filtering (normalized bandwidth $0.6\pi$ rad/s). The variables $T_s$ and $\alpha$ were kept fixed at 0.1 s and 1, respectively.
Figure 6.4: The effect low-pass filtering of the differentiated state vector, $\xi$, on parameter estimation for the Kalman filter based algorithm for data with $SNR = -9 \ dB$ (solid) and $SNR = 2 \ dB$ (dashed). The variables $T_s$ and $\alpha$ were kept fixed at 0.02 $s$ and 1, respectively.

Figure 6.5: The effect low-pass filtering of the differentiated state vector, $\xi$, on the prediction error for the Kalman filter based algorithm for data with $SNR = -9 \ dB$ (solid) and $SNR = 2 \ dB$ (dashed). The variables $T_s$ and $\alpha$ were kept fixed at 0.02 $s$ and 1, respectively.
Figure 6.6: The effect of the sampling period, $T_s$, on parameter estimation for the Kalman algorithm (solid) and the RPEM (dashed). Low-pass filtering was not applied, but would possibly improve the performance of the Kalman filter based algorithm for small sampling periods. The variables $SNR$ and $\alpha$ were kept fixed at 10 $dB$ and 1, respectively.

Figure 6.7: The effect of the sampling period, $T_s$, on the mean square of the prediction error for the Kalman filter based algorithm (solid) and the RPEM (dashed). Low-pass filtering was not applied, but would possibly improve the performance of the Kalman filter based algorithm for small sampling periods. The variables $SNR$ and $\alpha$ were kept fixed at 10 $dB$ and 1, respectively.
between the discretization error and the problems caused by differentiation of the noise. For the RPEM the choice of sampling period has a different effect. Not only does the RPEM perform better in general than the Kalman filter based method but it does not appear to have a performance optimum, at least not corresponding to the one of the Kalman filter based algorithm. The RPEM is less sensitive to small sampling periods, as there is no differentiation of the measured output. Hence, the sensitivity related to differentiated noise does not appear. Further, the regressors of the RPEM are adjusted in the adaption loop, which reduces the effect of discretization errors. The noise sensitivity of the Kalman filter based algorithm depends on the selection of $R_1$ and $R_2$. As usual, the measurement noise sensitivity can be reduced by relying more on the model i.e. by a decrease of $R_1/R_2$.

It would be easy to use the numerical example to draw the conclusion that the RPEM is superior to the Kalman algorithm. However, the Kalman algorithm suggested here is not meant to be an alternative to the RPEM, but rather a complement. As the output error model of the RPEM can have a non-convex criterion function, good initial parameters are crucial to obtain convergence to the global minimum. This is addressed next.

6.2.2 Application of the Algorithm for Initialization

The usefulness of the Kalman filter based algorithm for providing initial parameters to the RPEM is illustrated by identification of a second simulated system, previously described in [4]. The system, a temperature servo with nonlinear dynamics due to the energy loss caused by resistance, can be described by

$$\dot{x} = \begin{pmatrix} x_2 \\ -x_2 + g(u, x_1) \end{pmatrix} \quad y = x_1$$

(6.15)

where $g(u, x_1) = u - x_1 + (u - x_1)^2$. The input signal used to simulate the system was a PRBS-like signal with varying amplitudes (cf. [96]), and the model was discretized using a sampling period of 0.05s. No scaling was applied to the sampling period ($\alpha = 1$). When identifying (6.15) using the general model (3.15)-(3.16), it was exploited that the nonlinear function $g(\cdot)$ only depends on $x_1$ and $u$. By using this knowledge, and assuming that $x_2$ only appears as a linear term in $f(x, u, \theta)$, the number of unknown parameters in $\theta$ was reduced from 18 to 10.

The system described above was used to investigate the difference in performance of the RPEM when used with and without initialization by the Kalman filter based algorithm. In each case 100 runs using different initial vectors $\hat{\theta}(0)$ were the basis for the identification experiments. The data set, however, was the same in all runs. The initial vectors, $\hat{\theta}(0)$, in the experiments were uniformly distributed in a hypercube of dimension $n_\theta$, with the center in $\theta_{\text{stable}}$ and with a side equal to $2\sqrt{18}$, where

$$\theta_{\text{stable}} = (0 0 - 3 - 3 0 0 0 0 0 0)^T.$$  \hspace{1cm} (6.16)
Parameters were drawn from the hypercube and tested for stability. The first 100 parameters that corresponded to a stable linearized system were included in the set of initial vectors used in the experiments.

The idea of using these initial parameter vectors was to illustrate the performance of the algorithms under realistic conditions for a case where the true parameters are very poorly known. In such a case the initial parameters may be far away from the true parameter vector, and convergence to local suboptimal minima of the criterion function, or divergence, may occur for a substantial fraction of the runs. As a consequence, at least some initializations that resulted in correct convergence would be close to other initializations which would result in ill-convergence. Such border line initializations can be expected to have prolonged transient convergence time. Hence it would not be safe to assume that white residuals would be a good measure of correct convergence. Furthermore, measuring the whiteness of residuals is less well suited for output error methods, that are capable of handling colored noise. For this reason the evaluation of the convergence performance is, in the present example, based on (6.13)

For the experiments where initialization was used the initial Hessian of the RPEM was \( R(0) = 10^{-3}I \), and for the experiments without initialization by the Kalman filter based algorithm the initial Hessian of the RPEM was \( R(0) = I \). Different tuning was necessary in the two cases since when no initialization was used the magnitude of the Hessian of the RPEM needed to be high enough for the algorithm to converge from randomly chosen initial parameters. On the contrary, in the case where initialization by the Kalman filter based algorithm was used the initial parameters of the RPEM were assumed to be relatively close to the true parameter vector

\[
\theta_0 = (0 \ 1 \ 1 \ -1 \ -1 \ 2 \ 0 \ 1 \ 0 \ 0)^T,
\]

and hence a lower magnitude of the Hessian is required to prevent the RPEM algorithm from moving away too much from the initial values. No scaling was applied to the sampling period. The Kalman filter based algorithm was initialized with \( R_{1,x} = 10^{-8} \), \( R_{1,\theta} = 10^{-10} \), \( R_2 = 10^{-4} \), and \( P(0) = 10^{-1} \).

Out of the 100 different initialization vectors, the RPEM converged for all 100 experiments when used in combination with the initialization algorithm, whereas only 70 percent produced feasible results for the RPEM when used without initialization. In the remaining 30 experiments the RPEM algorithm did not converge to parameters corresponding to stable models. It is possible that some of the convergence problems could be avoided by individual tuning for each experiment, but this more detailed investigation is left for future research. The different initializations may have some impact on the algorithm performance, but the fact that the success rate increases from 70% to 100% when the proposed Kalman filter based algorithm was used for initialization, is so clear, that the results are deemed to be significant, despite the different initializations. It is stressed that this is highly important in critical engineering applications when malfunction of an algorithm cannot be accepted.

Another aspect of the performance with and without initialization is of course how large the error in the obtained parameter vector is. In Fig. 6.8 the number of experiments that fall below a certain relative error (6.13) in
the parameters is shown as a function of the relative error in the parameters. If e.g. the parameters can be allowed to have an error of 10% there are 19 experiments with initialization that satisfies this constraint, whereas only 8 of the experiments without initialization has an error of less than 10%.

When used for simulation the obtained models corresponding to correct convergence all perform very well. The MSE of the prediction error is more or less constant at $7.8 \cdot 10^{-5}$ for all the experiments, regardless of initialization. It appears that for this example the simulation performance was similar in all the successfully identified models. However, as the RPEM used without initialization diverged in 30 of the 100 cases, the conclusion remains that the method is more robust when used in combination with the Kalman filter based algorithm, which motivates the use of the Kalman filter based algorithm as an initialization method for the RPEM.

### 6.2.3 Results with Live Data

From the simulated examples it appears that the Kalman filter based algorithm can provide a useful initial parameter vector for the RPEM. To test this result further, measured data from a laboratory process were used for identification experiments.

The laboratory process on which the experiments were performed consists of two cascaded tanks with free outlets. The outlet of the upper tank feeds the lower tank whereas the upper tank is fed by a pump. In this type of system the water flow from each tank depends on the pressure at the outlet.
which in turn is a function of the water level. The water level in each tank is
measured using water level sensors that provide the water level given in volts.
The voltage can then be converted to centimeters by a scale factor (0.5 V/cm).
The input signal is the voltage to the pump, which operates in the range 0-10 V.
The input signal and the water level measurements are processed using a
PC equipped with MATLAB interfaces to the A/D and D/A converters.

Bernoulli’s principle and mass balance equations for the two tanks give the
following system description

\[
\begin{align*}
\frac{dh_1}{dt} &= -\frac{a_1\sqrt{2g}}{A_1} \sqrt{h_1} + \frac{k}{A_1} u \\
\frac{dh_2}{dt} &= -\frac{a_2\sqrt{2g}}{A_2} \sqrt{h_2} + \frac{a_1\sqrt{2g}}{A_1} \sqrt{h_1}
\end{align*}
\]

(6.18)

Here \( h_1 \) and \( h_2 \) are the water levels (in cm) in the upper and lower tanks
respectively. \( A_1 \) and \( A_2 \) are the areas of the horizontal cross section of each
tank, while \( a_1 \) and \( a_2 \) are the areas of the outlets. Further, the conversion
factor from input voltage \( u \) to input flow is denoted \( k \), and \( g \) is the gravity.
The process is clearly nonlinear, which makes it suitable for illustration of the
differences of the algorithms studied in the paper.

Data was collected from the system using a PRBS like input signal, but with
varying amplitudes (cf. [97] and Fig. 6.10). The system was then identified
using a second order model with one input. Consequently the model regressor

\[
\varphi(x, u) = (1 \quad u \quad x_2 \quad x_2u \quad x_1 \quad x_1u \quad x_1x_2 \quad x_1x_2u)^T
\]

(6.19)
Figure 6.10: Input signal used for cascaded tank experiments. The signal corresponds to the voltage to the pump that feeds the upper tank.

was obtained. Further the sampling period was chosen to 4\,s, and the scaling of the sampling period, $\alpha$, was 0.2. The Kalman filter based algorithm parameters $R_1, R_{1,\theta}$ and $R_2$ were chosen as $10^{-6}$, $10^{-8}$ and 1 respectively, the initial parameters $\hat{\theta}(0) = 0$ and $P(0) = 10^2$.

The Kalman filter based algorithm produced a parameter vector (for the scaled problem)

\begin{equation}
\hat{\theta}(7500T_S) = (-0.0002 \quad 0.0110 \quad -0.0992 \quad 0.0045 \quad -0.0043 \quad 0.0003 \quad 0.0077 \quad -0.0117)^T \quad (6.20)
\end{equation}

which was then used as initial parameters for the RPEM. In the RPEM the initial Hessian was chosen as $R(0) = 10^3I$, where $I$ is the identity matrix. Further, the stability limit of the projection algorithm was selected as $\delta = 0.01$. Here $1 - \delta$ is the maximal allowed pole radius of the linearized system. This limit was introduced to compensate for possible differences between the poles of the linearized system and the true nonlinear system stability (cf. [97]). At the end of the run with the RPEM algorithm the scaled ($\hat{\theta}$) and rescaled parameters ($\hat{\theta}_{\text{rescaled}}$) were

\begin{equation}
\hat{\theta}(7500T_S) = (-0.0008 \quad 0.0128 \quad -0.1036 \quad 0.0008 \quad -0.0065 \quad 0.0019 \quad -0.0088 \quad -0.0093)^T \quad (6.22)
\end{equation}

\begin{equation}
\hat{\theta}_{\text{rescaled}}(7500T_S) = (-0.0000 \quad 0.0005 \quad -0.0207 \quad 0.0002 \quad -0.0003 \quad 0.0001 \quad -0.0018 \quad -0.0019)^T. \quad (6.23)
\end{equation}
6.2. Experiments

Figure 6.11: Measured (solid) and simulated (dashed) output for the tank model obtained by the Kalman filter based algorithm. The dash-dotted line corresponds to the residual error.

For details on the conversion between scaled and rescaled (original) parameters see (3.21) and [96].

Simulations using the parameters obtained by the Kalman filter based algorithm gave reasonable results, see Fig. 6.11. The simulated output contained no major bias errors and followed the main variations of the measured output. The MSE of the prediction error was 0.16. This can be compared with the signal power of $y_m$ which was 11.36. However, the RPEM generated parameter estimates suitable for very high accuracy simulation of the system, as is shown in Fig. 6.12. The MSE of the prediction error using these parameters was only 0.06, or approximately 0.5% of the output signal power. Hence 99.5% of the signal power is explained by the estimated model. Note that it is the signal error power, which is penalized by the RPEM criterion. This figure is believed to be sufficiently high to indicate correct convergence of the combined algorithm. Further evidence is provided by the mean value of the residuals, which is $6.1 \cdot 10^{-3}$, or about 0.1% of the mean signal level. The autocorrelation function of the residuals, see Fig. 6.13, also support the fact that the system is accurately modeled. This follows since the autocorrelation function decays to negligible levels in approximately 35 samples. Comparing this to the sampling period of 4 s and the dominating time constants of the system, which is of the order 100 s, it follows that the main effect of the unmodeled dynamics are located within the transient after an input signal change.
Figure 6.12: Measured (solid) and simulated (dashed) output for the tank model obtained by the RPEM. The dash-dotted line corresponds to the residual error.

Figure 6.13: Autocorrelation function for the residual errors from the RPEM as a function of the time lag $\tau$. 
6.3 Initialization Software

The algorithms of Chapters 3 and 6 have been implemented as a MATLAB software package. The software was first developed for handling dynamic and static versions of the RPEM algorithm but have later been extended to include also the Kalman filter based initialization algorithm.

The focus of this chapter is on the new parts of the software package that enables the use of the initialization algorithm; by itself or in combination with the RPEM. For compatibility reasons and to simplify the use of the Kalman filter based initialization algorithm and RPEM together, as many files as possible have been reused from the RPEM case.

6.3.1 RPEM Software Package Overview

The MATLAB implementation of the initialization algorithm is part of a software package [99] containing all the files required to run the RPEM algorithm of Chapter 3, and an algorithm for identification of static systems. A detailed overview is also available. The software is designed to generate data for experimental purposes, as well as handling of live data, identification, and finally display of the results. Figure 6.14 shows a schematic picture of the files required to run the RPEM. Note that this picture was originally generated for a previous version of the software package, and does consequently not include the new functions needed for the initialization algorithm.

6.3.2 Initialization Software Overview

The software package is command driven, i.e. no graphical user interface is available. The package consists of a number of MATLAB scripts and functions, and has been developed to fit into the framework of the recursive identification software based on the RPEM described above. The new files can be divided into two groups (see Fig. 6.15)

1. Initialization: This group of m-files parallels the group Recursive identification of the RPEM software, but for the initialization case. The files h.m.m, dhdx.m.m, and reInitiate.m are in fact the same as for the RPEM algorithm. The new addition consists of a setup file SetupInit.m, and the Kalman filter based initialization algorithm Init.m.

2. Supporting functions: Again this group parallels a group (with the same name) of the RPEM software. The functions of this group are required to run the initialization algorithm but are not called by the user and does not need to be manipulated. The files dfdx.m.m and GenerateIndices.m are the same as in the RPEM software. Two new files F.m.m and differentiateY.m have been introduced here.

The remaining dashed blocks of Figure 6.15 represent groups of functions used for recursive identification as illustrated by Figure 6.14. Most of these functions can also be used in combination with the initialization algorithm. For example the data and setup parameters from the Live data measurements and
Figure 6.15: Overview of files for initialization.
Simulated data generation blocks can be used with the initialization algorithm in exactly the same way as they were previously used with the recursive identification block of the RPEM software. Several of the scripts and functions for preparation and display of results like SimulateModelOutput.m, PlotParameters.m, and PlotSystemAndModelOutput.m are also directly applicable to the output of the initialization algorithm. A comprehensive description of the files appear in Appendix B together with examples on how to run the software package.

6.4 Summary

In this chapter a method for identification of a nonlinear state space ODE was proposed. The method, a Kalman filter based algorithm, was developed for initialization of the RPEM of Chapter 3, which sometimes may converge to false suboptimal minima of the criterion function. The regressors of the proposed algorithm are obtained by numerical differentiation of the measurements. The algorithm allows for standard numerical differentiation using the Euler method, as well as for more elaborate differentiation schemes. As a consequence the resulting Bayesian algorithm does not converge to sub-optimal minimum points. The price paid is that the estimates may become biased. However, as stated above, the proposed algorithm can with advantage be used for initialization of the RPEM to reduce the risk of the RPEM converging to a false optimum. A final important advantage is that the proposed method can exploit scaling of the sampling period. This scaling method can improve the conditioning of the identification problem with several orders of magnitude.

Further contributions include a characterization of the accuracy aspects of the proposed algorithm and the RPEM. This investigation addresses noise sensitivity and the effect of the sampling period. Using the proposed Kalman filter based algorithm for initialization increased the success rate of the RPEM algorithm from 70% to 100% for a specific simulated example. The fact that success was obtained in all runs is crucial for systems where algorithmic malfunction is not acceptable. Further, the initialization algorithm has been integrated in the software package [99], which is available for download.

This chapter has only treated second order systems experimentally. For the Kalman filter based method higher order models imply repeated differentiation of measurement noise that would potentially cause problems in using the method as intended.
Chapter 7

MPC for Systems with Long Input Signal Dependent Delays

Model predictive control is common in industry today. The method is based on the use of a model to predict the consequences of different control choices. The advantage compared to many conventional nonlinear control strategies is that it enables the determination of a sequence of control signals, rather than the control at a single time instant, which may improve the overall performance. Further, it provides a way to prevent anticipated violation of e.g. signal constraints.

The control horizon is normally fixed, which creates problems when the method is to be used for systems with long and varying time delays. In this chapter such a problem, where the time delays of the system depend on the input signal, is addressed. The idea is to embed a controller designed for constant delays by input dependent sampling, and solve the optimal control problem inside the controller. The total control design is applied in an MPC setting.

7.1 Optimal Control of Systems with Constant Time Delay

A common way to incorporate time delays affecting inputs and disturbances in a discrete time state space model structure is to extend the state vector with delayed input signal samples from the present time, back to the expected delay of the input. For systems where the delay is large, this action introduces a high order model, which implies that significantly more computations may be required to find the desired control sequence. This should be particularly true in cases were constraint handling is required in the controller, and where general purpose software is used.

The use of another way of formulating the problem to avoid unneces-
sary computational complexity is discussed next. One main purpose with the present subsection is to formulate the two alternatives for discrete time optimal control, and to prove formally that they give equivalent results. After this, the chapter proceeds with details on the optimization algorithm, the implementation and the results. It can be noted that the treatment is valid for feedforward control when disturbances are present. Without disturbances the result also holds in the feedback case. Measured outputs are then used in the algorithm whenever possible. In Section 7.1, a shift of time scales is applied, in that $t + T_S$ is replaced by $t + 1$. This aligns the presentation to the literature and simplifies the notation in the proof.

7.1.1 Two Optimal Control Problems

The formal result to be proved is a generalization of the results in [98], to the case with multiple input signals, multiple output signals, and multiple measurable disturbances. The following multiple input multiple output (MIMO) discrete time nonlinear model is the basis for the analysis

$$
\begin{align*}
\dot{x}(t+1) &= f(x(t), u_1(t-T_{u_1}), \ldots, u_k(t-T_{u_k}), v_1(t-T_{D_1}), \ldots, v_\ell(t-T_{D_\ell})), \\
\hat{y}(t) &= Cx(t).
\end{align*}
$$

(7.1) (7.2)

Here $t$ denotes discrete time, $x(t)$ is the $(n \times 1)$ state vector, $u(t) = (u_1(t) \ldots u_k(t))^T$ is the $(k \times 1)$ input vector, and $v(t) = (v_1(t) \ldots v_\ell(t))^T$, $(\ell \times 1)$ are measurable disturbances. Further $T_{u_i} \in \mathbb{N}, i = 1, \ldots, k$, and $T_{D_j} \in \mathbb{N}, j = 1, \ldots, \ell$ denote the constant time delays associated with input $i$ and disturbance $j$ respectively. The right hand side of the nonlinear difference equation is described by the $(n \times 1)$ function $f(\cdot, \cdot, \cdot)$, $\hat{y}(t)$ contain the $(m \times 1)$ output signals while $C$ is the $(m \times n)$ output equation matrix. The problem studied includes feedforward, and it is hence assumed that $v(t)$ is measurable.

Extended State Vector Based Criterion

A common way to include time delays in discrete time state space models is to extend the state vector with delayed samples of the input signal. For a system with $k$ inputs the extended state vector can be defined as

$$
\bar{x}(t) = \left( x^T(t) \quad z_{1,1}(t) \quad \ldots \quad z_{1,T_{u_1}}(t) \quad \ldots \quad z_{k,1}(t) \quad \ldots \quad z_{k,T_{u_k}}(t) \right)^T,
$$

(7.3)

where

$$
z_{i,j}(t) = u_i(t + (j - 1) - T_{u_i}), \quad i = 1, \ldots, k, \quad j = 1, \ldots, T_{u_i}
$$

(7.4)

For simplicity it is assumed that the time delays are integer multiples of the sampling period. The state equation for the extended state vector can then be written as
\[ \ddot{x}(t + 1) = \]
\[
\begin{pmatrix}
  \dot{f}(x(t), z_{1,1}(t)), \ldots, z_{k,1}(t), v_1(t - T_{D_1}), \ldots, v_\ell(t - T_{D_\ell}) \\
  z_{1,2}(t) \\
  \vdots \\
  z_{1,T_{\epsilon_1}}(t) \\
  0 \\
  \vdots \\
  z_{k,2}(t) \\
  \vdots \\
  z_{k,T_{\epsilon_k}}(t) \\
  0
\end{pmatrix}
\]
\[ + 
\begin{pmatrix}
  0 & \cdots & 0 \\
  0 & \cdots & 0 \\
  \vdots & \vdots & \vdots \\
  0 & \cdots & 0 \\
  0 & \cdots & 0 \\
  1 & \cdots & 0 \\
  \vdots & \vdots & \vdots \\
  0 & \cdots & 0 \\
  0 & \cdots & 1
\end{pmatrix} u(t) =
\begin{pmatrix}
  \dot{f}(x(t), z_{1,1}(t), \ldots, z_{k,1}(t), t) \\
  z_{1,2}(t) \\
  \vdots \\
  z_{1,T_{\epsilon_1}}(t) \\
  0 \\
  \vdots \\
  z_{k,2}(t) \\
  \vdots \\
  z_{k,T_{\epsilon_k}}(t) \\
  0
\end{pmatrix}
\]
\[ + 
\begin{pmatrix}
  0 & \cdots & 0 \\
  0 & \cdots & 0 \\
  \vdots & \vdots & \vdots \\
  0 & \cdots & 0 \\
  0 & \cdots & 0 \\
  1 & \cdots & 0 \\
  \vdots & \vdots & \vdots \\
  0 & \cdots & 0 \\
  0 & \cdots & 1
\end{pmatrix} u(t) \triangleq \ddot{f}(\ddot{x}(t), t) + Bu(t). \quad (7.5)
\]

Note that since the disturbance vector is measurable \( v_i(t - T_{D_i}), \ i = 1, \ldots, \ell \) are known time variables of the right hand side of the difference equation, and can therefore be replaced by numerical values when evaluating (7.5). This is the reason why \( v_i(t - T_{D_i}) \) is excluded in the right of the third equality. Using the extended state vector, the corresponding output equation becomes
\[
\dot{y}(t) = (C \ 0)\ddot{x}(t) \quad (7.6)
\]
Next define the \((k \times (n_x - n_x))\) matrix
\[
B_1 = \begin{pmatrix}
1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\
0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0
\end{pmatrix}.
\] (7.7)

From (7.5)-(7.6) a standard optimal control criterion with arbitrary penalties on the control errors and inputs (now parts of the extended state vector) can be formulated as
\[
J_{\text{extended}} = \sum_{t=0}^{N-1} \left( \| y_{\text{ref}}(t) - (C \ 0)x(t) \|_{Q_1}^2 + \| \text{vec}(z_{i,1}(t)) \|_{Q_2}^2 \right)
= \sum_{t=0}^{N-1} \left( \| y_{\text{ref}}(t) - (C \ 0)x(t) \|_{Q_1}^2 + \| (0 \ B_1)x \|_{Q_2}^2 \right),
\] (7.8)

where \(\text{vec}(\cdot)\) denotes a matrix of stacked row vectors. The vectors to stack are given by \(i\). Furthermore
\[
\| s(t) \|_{Q_1}^2 = s^T(t)Qs(t)
\] (7.9)

for some signal \(s(t)\) and the square symmetric positive semi–definite weighting matrix \(Q\). Specifically \(Q_1\) is a \((m \times m)\) matrix containing the penalties for the errors on each output, whereas \(Q_2\) \((k \times k)\) penalizes the delayed inputs \(z_{i,1}(t)\). The minimization of (7.8) is performed subject to the dynamical constraints (7.5) and (7.6). Note that the optimization algorithms applied to solve (7.8), need to compute controls outside the horizon that appears explicitly in the criterion.

**State Prediction Based Criterion**

A second approach is to use the delayed inputs \(u_i(t - T_{u_i})\) in the criterion without the use of the extended state vector. The optimal control criterion then becomes
\[
J_{\text{direct}} = \sum_{t=0}^{N-1} \left( \| y_{\text{ref}}(t) - Cx(t) \|_{Q_1}^2 + \| \text{vec}(u_i(t - T_{u_i})) \|_{Q_2}^2 \right).
\] (7.10)

The minimization of (7.10) w.r.t. \(u_i(t - T_{u_i}), i = 1, \ldots, k\) is performed subject to the dynamic constraint (7.1) and (7.2). In this case the penalty on the control error needs to be computed by means of a forward prediction in the associated software, where the model is iterated ahead in time. The advantage as compared to the method that exploits state vector extension is that the forward prediction is performed by iteration of a lower dimensional dynamic system. This can be clearly seen in the linear case where the extended system matrix is filled with many zeros if the delay is large. In cases where standard software packages are used, these zeros can give rise to a large number of void operations that only cause an increase of the computational complexity. To
provide a firm ground for the choice of the direct criterion rather than the extended state vector criterion, the issue of the equivalence between (7.8) and (7.10) is next addressed formally.

### 7.1.2 Equivalence of the Two Control Problems

The analysis of the equivalence between the two criteria builds on the Euler–Lagrange equations, given by the following lemma

**Lemma 5.** Assume that \( L(x(t), u(t), t) \) and \( f(x(t), u(t), t) \) below are continuously differentiable with respect to \( x \) and \( u \), uniformly in \( t \). A necessary condition for the optimality of the performance index

\[
J = \sum_{t=0}^{N-1} L(x(t), u(t), t)
\]  

subject to the dynamics

\[
x(t + 1) = f(x(t), u(t), t)
\]  

is then that the Euler-Lagrange equations

\[
x(t + 1) = f(x(t), u(t), t)
\]  

\[
\lambda(t) = L^T_x(x(t), u(t), t) + f^T_x(x(t), u(t), t)\lambda(t + 1)
\]  

\[
x(0) = x_0
\]  

\[
\lambda(N) = 0
\]  

\[
L_u(x(t), u(t), t) + \lambda^T(t + 1)f_u(x(t), u(t), t) = 0
\]  

hold.

Here the Lagrange multipliers are \((n \times 1)\) vectors. The subscripts of \( L \) and \( f \) denote partial differentiation w.r.t. the variable in question. As before, the measurable disturbance is incorporated in \( L(x(t), u(t), t) \) through the time argument \( t \).

**Proof:** See e.g. [24].

Using Lemma 5, the following theorem can now be proven to hold

**Theorem 2.** Assume that

\( A_1 : \) \( f(x(t), u_1(t - T_{u_1}), \ldots, u_k(t - T_{u_k}), v_1(t - T_{D_1}), \ldots, v_l(t - T_{D_l})) \) of (7.1) is continuously differentiable with respect to \( x, u_i, i = 1, \ldots, k \), uniformly in \( v_i(t - T_{D_i}) \).

\( A_2 : \) \( Q_1 \) is symmetric positive semidefinite.

\( A_3 : \) \( Q_2 \) is symmetric positive definite.
Then the optimal control problems defined by the performance indices (7.8) and (7.10) have identical solutions. Moreover, the control is given by

\[
vec(u_i(t - T_u)) = \sum_{i=1}^{N-1} \left( \prod_{j=t+1}^{i-1} f^T_x(j) \right) C^T Q_1 (y_{ref}(i) - \hat{y}(i)),
\]

\[t < N - 1. \quad (7.18)\]

**Proof:** See Appendix A

### 7.1.3 Controller Design Algorithm

The optimization algorithm to solve e.g. (7.25)–(7.27) was selected as the straightforward gradient algorithm denoted 'DOP0' in [24]. Although more efficient schemes may exist, the implemented algorithm performed robustly and did not limit performance by any means. In the SISO case the algorithm is, with reference to Lemma 5

\[
\begin{align*}
\text{Initiate} \\
x(0), u(i - T_u), \ i = 0, ..., N - 2 \\
\text{Iterate} \\
x(i + 1) = f(x(i), u(i - T_u), i), \ i = 0, ..., N - 2 \\
\lambda^T(N - 1) = L_x(x(N - 1), u(N - 1 - T_u), N - 1) \\
H_u(i) = \lambda^T(i + 1) f_x(x(i), u(i - T_u), i), \ i = N - 2, ..., 0 \\
\lambda^T(i) = \lambda^T(i + 1) f_x(x(i), u(i - T_u), i), \ i = N - 2, ..., 0 \\
\Delta u(i - T_u) = -k H_u^T(i), \ i = 0, ..., N - 2 \\
\Delta u_{mse} = \sqrt{\frac{1}{N-1} \sum_{i=0}^{N-2} \Delta u^T(i - T_u) \Delta u(i - T_u)} \\
\text{if} \ \Delta u_{mse} < \varepsilon \ \text{stop} \\
u(i - T_u) \rightarrow u(i - T_u) + \Delta u(i - T_u), \ i = 0, ..., N - 2.
\end{align*}
\]

The reason why the SISO case is presented is that this is the case implemented in the software, which is used in Chapter 8. In the algorithm \(H\) denotes the Hamiltonian, see e.g. [24], the constant \(k\) determines the step size of the input signal update \(\Delta u(i - T_u)\), whereas \(\varepsilon\) defines the stopping criterion of the numerical search. If \(k\) is chosen too large the algorithm will not converge and if chosen too small the convergence will be slow and computationally demanding without necessarily achieving better control. The control penalty \(\rho\) acts on the derivative of the input and can be chosen larger for a smoother, slower control signal or larger for a faster control signal and associated higher bandwidth.
7.2 Input Dependent Sampling

Computers used to operate systems and controllers usually use regular equidistant time sampling. To compensate for the input signal dependent time delays the controller utilizes a different sampling period, dependent on the value of the input signal. This dependence is e.g. preferred in flow control applications where the time delay is proportional to the inverse of the input. The idea is that by utilizing this relationship and perform sampling at a rate that is proportional to the inverse of the input signal the time delay will become constant as measured in terms of the number of samples inside the controller. It would thereafter be straightforward to use models obtained from regular system identification methods with constant time delays. Note that only one of the input signals is assumed to affect the sampling rate. Generalizations are left for future research.

7.2.1 Algorithm

The values of each input and disturbance signal at the instant of internal sampling is obtained by interpolation of the measured signals from the system. If the internal sampling instant $t_S(i)$ lies between the system sampling instants $t_T(i-1)$ and $t_T(i)$ the value of the input signal at instant $t_S(i)$ can be calculated as

$$u(t_S(i)) = u(t_T(i)) - \frac{t_T(i) - t_S(i)}{T_S}(u(t_T(i)) - u(t_T(i-1)))$$  \hspace{1cm} (7.20)

Here $T_S = t_T(i) - t_T(i-1)$ is the sampling period of the system, and $u$ is the input signal that affects the sampling. Note that $t_T$ are the instants at which data is sampled from the system. Given the time delay there are, at any given instant, a sequence of data points available, from which the effect has not yet shown up in the output signal. This means that with an accurate model, predictions can be made about what will happen in the system in the near future. This information can be utilized when calculating a control sequence for the next few samples, to obtain an overall desirable behavior of a model predictive controller.

After calculating the desired control sequence for the internal sampling instants $t_S$ the same interpolation method can be used to calculate the input signal value at time $t_T(t+1)$ (the new control signal to be fed to the system). That is

$$u(t_T(t+1)) = u(t_S(k)) - \frac{t_S(k) - t_T(t)}{t_S(k) - t_S(k-1)}(u(t_S(k)) - u(t_S(k-1)))$$  \hspace{1cm} (7.21)

for some value of $k$, such that $t_S(k-1) \leq t_T(t+1) \leq t_S(k)$.

7.3 MPC

7.3.1 The MPC Principle

Model predictive control is usually based on optimal control, with a similar optimization criterion that is solved for the control $u$ [60]. Through the cal-
calculation of the optimal control variable for several time instants, the method is particularly useful for systems with limitations on the states and/or the inputs, and for systems with time delays. The MPC paradigm, proposed in [51] is straightforward to apply. At a certain time an open-loop optimal control problem is solved, based on the current state of the system. The obtained control variables are then used for a period of time, after which a renewed calculation is performed. The procedure is then repeated. MPC is today a standard tool in the process industry, see e.g. [27]. The basic idea is to use the measured data up to time $t$, and a model of the system in an iterative search for the control sequence $u(t + T_S) u(t + 2T_S) \ldots u(t + MT_S)$ that minimizes a criterion function, typically on the form

$$V(u) = \sum_{s=1}^{P} (y_{ref}(t + sT_S) - \hat{y}(t + sT_S))^2 + \sum_{s=1}^{M} \rho(u(t + sT_S))^2$$  \hspace{1cm} (7.22)$$

where $V(u)$ is the loss function, $y_{ref}(t)$ is the reference signal for the predicted output $\hat{y}(t)$, and $\rho$ is the penalty on the control signal $u$. The prediction horizon $P$ is the number of samples the prediction of the criterion covers, while the control horizon $M$ is the length of the control sequence. Consequently $M$ determines the size of the optimization problem, and hence the computational load and execution time when the controller is used on-line. Note that the equation (7.22) represents the single input single output (SISO) case without time delays, but the criterion can easily be generalized to the multiple input multiple output (MIMO) case. As compared to the HJB that solves the feedback problem exactly [25], MPC is far less complex; no discretization of states and input is e.g. required.

7.4 Optimal Feedforward MPC Software

This section contains a description of a MATLAB software implementation of an optimal controller designed for systems with input varying time delays. More specifically the input dependence is such that the delay is proportional to the inverse of the input. Such delays typically occur in systems where the input is a velocity or a flow, e.g. pipe flow, water reservoir systems or a vehicle in traffic. The input dependence is compensated for through a resampling (7.21) that makes the time delay appear constant as counted in the number of samples inside the controller. Thereafter it straight forward to use a conventional control scheme for time delayed systems. Here the controller is based on optimal control theory (7.19) implemented in an MPC framework.

7.4.1 Introduction

As stated before, the time delays of the system are assumed to depend on the control signal. It is further assumed that the model used for calculating the control does not take this variation into consideration. Consequently, the delays need to be compensated for outside the controller. As the controller was developed to be utilized in combination with a computer that samples regularly,
an input signal dependent resampling between the system and the controller can be used in order to create the effect of a constant time delay (in samples), as seen by the controller design algorithm. The details of this input signal dependent sampling has been described above. It is thereafter straight forward to use the dynamic description (7.25)–(7.27) in an optimization algorithm, where the variation of the input and disturbance signal delays have been compensated for. After calculating the optimal control sequence for the input sampled data points the information is re-transformed to the regularly sampled time instants at which the new control signal can be fed to the system. A block diagram of the architecture of the controller software is shown in Fig. 7.1. A more detailed description of the implementation can be found in [18]. The free software package can also be downloaded from the same address.

The optimization algorithm to solve e.g. (7.25)–(7.27) was selected as the straightforward gradient algorithm denoted 'DOP0' in [24]. The algorithm is given by (7.19).

This section is intended as a brief user manual for the package of MATLAB scripts and functions developed for feedforward optimal control/MPC of a system with input dependent time delays. A more complete manual for the code is available in [18]. The software was developed for control of solar collectors with attached heat storage tanks located in Seville, Spain, as described in Chapter 8. The objective is to keep the outlet temperature from the collectors as constant as possible despite variation in solar radiation and inlet temperature. The two latter signals are considered to be measurable disturbances, as neither can be manipulated in the Sevillian solar cooling plant. The input signal to the identified model is the flow through the solar collector. The idea is that the variations in solar radiation that appears on a partly sunny day can be, at least to some extent, compensated for by changing the flow accordingly. At the solar plant the flow is changed by manipulation of the speed of the pump that circulates the water through the collectors and tanks. The relation between the pump speed (given in % of full speed) and the flow through the system is assumed to be linear. Hence, the pump speed is the actual control signal of the system, but the model uses the scaled flow as input. The work of identifying the black-box
model of the solar collectors and tanks, which this software was designed to control, will be described in Chapter 8. The model structure and identification algorithms in Chapter 3 and 6 were used to generate the model that both the simulator and the controller utilizes.

The optimization algorithm (7.19), see [24], uses gradient search to find the optimum of a criterion function chosen by the user. To handle the varying time delays a resampling, depending on the input signal is performed inside the controller. By the use of this varying sampling period the time delay becomes constant in number of samples, and it is thereafter straight forward to compensate for the delay.

The code consists of two parts, the first part is a simulator representing the system. The second part is the controller algorithm, which computes the control to be applied to the simulated system.

In the simulator the model is based on

$$x(t + T_S) = x(t) + T_S \varphi^T(x(t), u(t - T_S T_u(u)), v_1(t - T_S T_D(u)), v_2(t - T_S T_D(u))) \theta$$

$$y(t) = x(t)$$

(7.23) (7.24)

where $\varphi$ is a regressor vector containing nonlinear elements that depend on the flow $u$, the disturbances $v_1$ and $v_2$, and the state, which according to (7.24) corresponds to $y$.

Inside the optimal controller the system has been reformulated to enable penalization of the derivative of the input, thereby achieving integral action. It is straight forward to fit the original model (7.23)-(7.24) into the notation of [24] by the use of the following extended state space representation

$$\ddot{x}(t) = (x(t) - u(t - T_s T_u))^T$$

$$\ddot{x}(t + T_s) = \ddot{x}(t) + T_s \left( \begin{array}{c} \varphi^T(\dot{x}(t), v_1(t - T_s T_D), v_2(t - T_s T_D)) \theta \\ \dot{u}(t - T_s T_u) \end{array} \right)$$

$$\dot{y}(t) = x_1(t).$$

(7.25) (7.26) (7.27)

By extending the state space model in this way, both $x$ and $u$ are included in the state vector, and the derivative of the input, $\dot{u}$, can be treated as an input. Note that $T_s$ denotes the internal (input dependent) sampling period of the controller. It is stressed that $T_s$ is held constant during the iterations performed by the control sequence optimization, and that it is only recalculated when a new control sequence is to be re-calculated according to the MPC principle. This simplification is introduced to avoid the need for dynamic buffer re-allocation within the optimization algorithm.

### 7.4.2 Optimization Horizon

Since the software has been developed to handle controllers with as well as without integral action, the criterion function of the optimal control problem in the code has been chosen as

$$J = \sum_{s=0}^{N-1} \left( (y_{ref}(t + sT_s) - Cx(t + sT_s))^2 + \rho u^2(t + sT_s - T_s T_u) \right).$$

(7.28)
By performing the integration for $u(t_1), t_1 < t$, the occurrence of large static errors may be reduced in the case without integral action. The errors originate from the fact that only the first element of the control sequence is fed to the system before a new sequence is computed. The topic is discussed in detail in [98]. For the case with integral action this is of less consequence, however, it may still affect transient integrating control performance.

### 7.4.3 Software Package Overview

The software package is command driven, i.e. no GUI is available. The package consists of a number of MATLAB scripts and functions. In this section the content of each function/script and their relations are described. A more comprehensive description of the files of the software package can be found in Appendix D.

As shown in Fig. 7.2 there are four main groups of m-files

1. **Setup simulator and controller variables**
   This group consists of one file only, SetupSim.m, which generates and initializes all the variables required to run the simulator and the controller.

2. **Simulator**
   This group also consists of one file, Simulator.m. The file simulates a system and calls the controller.

3. **Flow sampling and controller**
   The third group contains three files; flowSample.m, which resamples the data depending on the input signal, optimalController.m that contains the optimization algorithm, and controllerShell.m. The last file calls the other two, and is the only function called from the simulator.

4. **System description and optimization criterion**
   The last group contains the files updateS.m, dLdu.m, dLdx.m, f.m, df.m, and calculatePhi.m. These files describe the system properties and the optimization problem to be solved, and are chosen by the user. Each file is called by either optimalController.m or controllerShell.m (see Fig. 7.2).

### 7.4.4 Code Validation

A number of runs have been made to confirm that the code behaves as can be expected with respect to the input dependent sampling, the time delays and the controller.

**Input dependent sampling**

To verify the accuracy of the resampling, simulation experiments have been run where the system and model are linear in the input and states, but with
Figure 7.2: Schematic picture of the MATLAB functions.
nonlinear combinations of the two measurable disturbances affecting the process. This model was developed for control the outlet temperature of a field of solar collectors, and will be discussed in detail in Chapter 8. In Fig. 7.3 the internal sampling period inside the controller is plotted as a function of the input signal (flow). It can be seen that the sampling period is proportional to the inverse of the input, exactly as expected.

**MPC and time delays**

To verify the behavior of the controller, simulation experiments were performed using a linear system and model, originally developed for use with the same solar collectors as above. The reference signal was kept constant (see Fig. 7.4) throughout the experiment. The output and corresponding control signal can be seen in Fig. 7.4 and 7.5 respectively. The change in input signal at time 2000 s, corresponds to a change in the second disturbance signal (the inlet temperature) by approximately $3^\circ C$, as can be seen in Fig. 7.6. At times 3000 s and 5000 s the first disturbance signal (the radiation) changes with a corresponding change in flow as a result. The effect can be seen in the slightly delayed output signal, which verifies that the system time delay is handled correctly. The delays from the disturbances to the output are greater than (radiation) or equal to (inlet temperature) the delay from input to output. This means that it is at least theoretically possible to counteract part of the effect.
of the disturbances on the output. In Fig. 7.4 a small offset from the reference signal can be seen as the output settles after the changes in the disturbance signals, which indicates that the integral action on a finite control horizon will not necessarily guarantee a zero offset error. This points out a highly interesting future research task, since offset free MPC is of central importance.

**Nonlinear models**

To verify that the code performs as intended also for nonlinear systems and models a simulation run was performed using a nonlinear model of the solar collectors in the simulator and controller. The results can be seen in Fig. 7.7-7.9. Note that since neither the simulated system nor the model in the controller is the same as in Chapter 8, the results here are not directly comparable to those of Chapter 8. This is, however, not important, as the purpose of this example is to illustrate that the code is capable to handle also nonlinear models.

**7.5 Summary**

The chapter has treated optimal control, in particular MPC, for nonlinear systems with long and varying time delays. Two formulations of a MIMO optimal control problem were defined, one direct and one based on an extended state
7.5. Summary

Figure 7.5: Input signal for control simulations with a linear model and system.

Figure 7.6: Disturbance signals for control simulations with a linear model and system.
Figure 7.7: Output (solid) and reference (dashed) signal for control simulations with nonlinear model and system.

Figure 7.8: Input signal for control simulations with nonlinear model and system.
vector with delayed inputs added as state variables. The control problem setting is for constant time delays. It was then proved that the two corresponding optimal control problems give the same optimal control sequence. This is important since a criterion with state extension can lead to very high orders for long time delays, a fact that may be prohibitive for real time implementation of MPC schemes. The solar powered cooling plant to be discussed later, e.g. required delays of more than 10 sampling periods. The time variations were handled by the introduction of input dependent sampling. Together with the optimal control formulation, this defines a new way of handling the considered set of control problems, in an MPC setting.

Figure 7.9: Disturbance signals for control simulations with nonlinear model and system.
7. MPC for Systems with Long Input Signal Dependent Delays
Chapter 8

Identification and Control of a Solar Cooling Plant

8.1 Introduction

The present chapter discusses optimal feedforward model predictive control (MPC) of solar collector plants. Solar collector plants can e.g. be used for power generation, heating and air-conditioning [10; 11; 43]. The technology is useful also in small scale systems, e.g. for residential homes. Very often, the fluid temperature at the outlet of the solar collector field needs to be controlled, to avoid boiling and to provide the correct operating temperature for the systems that use the heated fluid as the source of energy.

In the present study, the control objective is to keep the outlet water temperature (the output) of the solar collector field at a constant level, in order to optimize the performance [48] of an experimental 35 kW air-conditioning system located at the University of Seville, Spain [100]. The principles of the cooling machinery are further described in e.g. [75], [12] and [101]. That part of the plant is outside the scope of the thesis. The available input signal is the pump speed which is directly proportional to the flow of water through the plant. The outlet water temperature is strongly affected by two measurable disturbances, these being the solar radiation and the water temperature at the inlet of the solar collector field. A feedforward control strategy is therefore needed. This control problem poses a number of challenges. First, there are strong reasons to believe that the plant is subject to nonlinear effects, due to the physical heat absorption and radiation processes. Black body radiation is e.g. dependent on the fourth power of the absolute temperature [45]. Secondly, the time delays of the plant are long. The pump speed only affects the output temperature after replacement of the major part of the water stored in the solar collector field. The inlet temperature is subject to similar pure transport delay effects, whereas the solar radiation is also distributed [26]. Most importantly, the delays are strongly dependent on the flow. In addition to the strong
flow dependence of the delays, also the transient time describing flow to output temperature is expected to be subject to similar effects, whereas the transient from solar radiation is probably more constant.

In feedforward control it is crucial that an accurate model is available. With such a model the remaining error that needs to be compensated for by feedback is reduced. This is useful e.g. for increasing the margins of the feedback controller, or for increasing the bandwidth even further. To obtain a nonlinear dynamic model that describes the solar collector plant, nonlinear system identification applied to measured data is used in the present chapter. The selection of identification method for the present work is motivated by the the selected controller design method which requires a continuous time differential equation model of high accuracy. An algorithm for identification of continuous time nonlinear systems is therefore advantageous to use. Discretization errors can then be directly compensated for by the adaptation loop, which is not the case if a post-transformation from discrete time is performed. Furthermore, since the objective is model accuracy, output error identification algorithms are known to provide the best accuracy [87], and are hence the preferred choice. Finally, due to the difficulty to model the non-linear distributed effects [26] of the plant from first principles, a black-box identification algorithm is needed. The above requirements are all met by the joint application of the nonlinear identification algorithms presented in Chapter 3 and 6.

The main scope and overall contribution of the present chapter is the development of a new non-linear optimal feedforward controller of MPC type for control of solar collector plants. Data from the plant of the University of Seville was used for system identification and subsequently to design the controller and to assess the achievable performance. In order to highlight the performance achievable by feedforward, no feedback or constraint handling is included in the present work. It is understood that these components need to be added in a commercial system, however this is left for future research. The controller design methodology is based on Chapter 7. The identified continuous time differential equation model, extended with an additional integrator is used for the design, to allow accurate regulation over large ranges. Flow dependent sampling is a prerequisite for the application of the nonlinear optimal control algorithms that operate on plant models with constant delays. A further effect of this strategy is that the transients are affected. Of these transients, the transient model from solar radiation to output temperature may be negatively affected by this, however the impact of this effect is expected to be significantly smaller than the effect of the flow dependent delay. For this reason the effects on the transients are neglected in this work and left for future research.

8.2 The Solar Powered Air–conditioning Plant

8.2.1 Plant Overview

The solar air conditioning plant studied in this chapter is located in Seville (Spain) and is used to acclimate the Laboratories of the System Engineering and Automation Department of the University of Seville. The plant can operate
8.2. The Solar Powered Air-conditioning Plant

for cooling in the summer and heating in the winter. It consists of a solar collector field that produces hot water, which, in the cooling case, is fed to an absorption machine that generates chilled water and injects it into the air conditioning system. The cooling power achieved is 35 kW. In the heating case, the hot water from the solar collectors is fed directly to the air distribution system.

A general scheme of the plant is shown in Fig. 8.1, showing its main components: the solar collector system, composed of a set of flat solar collectors; the accumulation system, composed of two tanks storing hot water; and the cooling machine. There is also an auxiliary gas-fired heater that can supply energy in situations where the available solar radiation is insufficient; and a load simulator (a heat pump) that allows for performing tests for different load profiles.

![Solar cooling plant scheme](image)

Figure 8.1: Solar cooling plant scheme.

In both the heating and the cooling case, the overall control objective is to supply water to the air distribution system at the demanded temperature minimizing auxiliary energy (gas) consumption. Also, the stored energy in the tanks at the end of the day is of importance, since it can be used in the following morning when the solar radiation is low. The primary energy source (solar radiation) cannot be manipulated and has to be treated as a measurable disturbance. The inlet water to the absorption machine in the cooling case, or air distribution system in the heating case, is a mix of the water coming from the solar collectors, the storage tanks and the water coming from the gas-fired heater if the additional energy source is used. Additionally, the temperature
of the water in the solar collectors can be controlled adjusting the water flow inside the solar field.

In the refrigeration configuration, operational constraints in the absorption machine must be satisfied. This implies that the control system must keep the cooling machine working at the desired operating point and this is achieved by keeping the machine inlet water temperature at a given constant set-point, see [48], p. 714 for the motivation.

The plant operates with two different energy sources (solar power and gas), which can be combined or used independently. In addition, thermal energy coming from a storage tank can be added to the system. The plant can be re-configured on-line manipulating valves (open/close) and pumps (on/off) to allow for selection of the components for energy supply.

8.2.2 Solar Collector Dynamics

The solar collector field of the solar cooling plant is a set of flat-plate collectors, see Fig. 8.2, each of which consists of an absorber. Water passes through pipes located below the absorber plate, and is thereby heated. This style of collector, although inferior in many ways to evacuated tube collectors, is still the most common type of collector in many countries. Flat collectors demonstrate a good price-performance ratio, as well as a broad range of mounting possibilities. In

Figure 8.2: Flat solar collector field located at the University Seville.
[9], the results of the cost analysis showed that the best performance is achieved with common flat plate solar collectors driving a single effect absorption chiller.

In this chapter the focus is on control of the output temperature of the solar collectors, which is the main source of energy of the plant. If the collectors are operated adequately the required use of the gas heater will decrease. In addition the absorption machine shows higher efficiency [48] if the water fed to it holds a relatively constant temperature.

From a control perspective the solar collectors show several interesting and challenging effects. To begin two of the main components affecting the collector outlet temperature, the solar radiation and the inlet temperature, must be treated as disturbances. This means that the operation of the system is largely dependent on factors that can be measured but not chosen freely. In addition, the solar collector dynamics is subject to nonlinear effects as well as to large and flow variant time delays. With normal selection of the sampling period, the delays are at least 10 sampling periods and may vary by more than a factor of 5 with normal variations of the controlling flow. The dynamic behavior shows similarities with a first order linear system, with one dominating transient.

8.2.3 Control Objective

The solar collectors were studied in combination with the accumulation tanks. The objective was to keep the outlet temperature from the solar collectors as close as possible to a reference temperature despite variations in solar radiation due to the weather being partly cloudy. This objective was motivated by the fact that the cooling effect obtained by the absorption machine is higher and much more reliable if the temperature fed to it is constant or shows small variations [48]. It can be noted that a constant temperature is also an advantage when heated water is used for other purposes. Examples include e.g. standard power generation and desalination, see [28] for further details. The use of the accumulation tanks makes the inlet temperature unaffected by the rapid changes in radiation, as the tanks will smooth out the effect of outlet temperature variations before the water is fed back to the collectors. The flow through the collectors is a natural choice of control signal, as e.g. the change in outlet temperature caused by a decrease in radiation can be compensated for by reducing the flow, and thereby allowing the water more time in the collector when the heating effect is lower. This in turn implies that the time delays of the system depend strongly on the control signal. Some, but not all, transients are also believed to show a similar behavior.

8.3 Nonlinear System Identification

8.3.1 Identification Method Selection

There are a number of model based controller design methods for nonlinear systems, see [44] for some alternatives. Often the design is based on a nonlinear ordinary differential equation (ODE) model. When performing system identification it can therefore be advantageous to estimate a continuous time model
of the system. If this is done within the algorithm, the effect of discretization errors are e.g. compensated for by the adaptation loop [97], as compared to when a transformation to continuous time is performed after system identification. In this chapter, the nonlinear dynamic model is used for feedforward control of the outlet temperature of the solar collector, a fact which requires the model of the system to be accurate. It is well known [87] that an output error method is then preferable for identification. The drawback is the possibility for convergence of the algorithm to sub-optimal minima of the criterion function, something that can be countered by an initialization algorithm, run before the main identification algorithm. For the above reasons the RPEM described in Chapter 3 and the initialization algorithm developed in Chapter 6 were used in the present work. It is however stressed that other methods, e.g. from the NARMAX class, could be applied as well. A comparison between different identification methods is beyond the scope of the thesis.

8.3.2 Requirements on Measurements

To identify a reliable nonlinear model it is important to collect data using inputs that contain a wide variety of amplitudes as well as of frequencies [93]. This implies that the flow needs to be varied during the data collection. In addition it is recommendable to collect data on a day with partly cloudy weather, to enable extraction of as much information as possible on how the solar radiation affects the outlet temperature dynamically. It would of course be desirable to study the impact of the inlet temperature in a similar way, but due to the way the system is designed it would be impossible to obtain high frequency variations in the inlet temperature without simultaneously introducing problems associated with identification in closed loop. The effects of high frequency variations in the inlet temperature are therefore not treated in this chapter. During the measurement campaign preceding the identification experiments described later in this section, there was one day that fulfilled the requirements above. With a sampling period of 5 s, the total data set amounts to approximately 3500 samples. This should be sufficient for fitting a typical model parameterized by approximately 10 parameters.

As can be seen in Fig. 8.3, the units used to measure the flow, the inlet and outlet temperatures, and the solar radiation differ by several orders of magnitude. It is therefore necessary to scale the signals before performing identification experiments. Here the scaling was chosen so that the variance of the scaled outlet temperature, the scaled radiation and the scaled flow measured over the whole data set was 1 for each of three signals. As the inlet temperature shows small variations it was not treated separately, but scaled in the same way as the outlet temperature. The scale factors that each signal was multiplied by to obtain the unit variance was

\[
\begin{align*}
\beta_{temperature} &= 0.1598 \\
\beta_{flow} &= 6.5340 \cdot 10^{-4} \\
\beta_{radiation} &= 4.9515 \cdot 10^{-3}.
\end{align*}
\]

The subscripts indicate what signals each scale factor was used for.
8.3.3 Off-line Time Variable Delay Compensation of the Data

As pointed out earlier the delays associated with the time it takes the heat transporting fluid to travel through the pipes of the solar collectors depends on the flow through the collectors. Therefore, in order to fit data from the system to a standard continuous time model structure these varying time delays need to be taken into consideration. For a system with a constant time delay, this could be achieved by shifting the inputs accordingly. In order to make the solar collector data applicable to the standard methods, however, the shifting needs to be done depending on the flow.

In the system there are flow variant delays acting on the radiation, the inlet temperature, and the flow. For example it will take longer before the effect of a step change in the flow can be seen in the outlet temperature if the step is taken from 500 to 1000 \(L/h\), than if the step is taken from 2500 to 3000 \(L/h\). The limited variations in the inlet temperature makes it very difficult to estimate the delay on this signal, but the other two delays are more distinct in the data set. By examining the data manually the time delays associated with flow and radiation were estimated for different values of the flow through the collectors. For example, when a cloud passes the sun there is a dip in

---

Figure 8.3: Measured data from the solar collector plant. From top to bottom the figure displays the outlet temperature, the solar radiation, the inlet temperature and the flow.
the radiation. A while later there will be a corresponding dip in the outlet temperature from the solar collectors, see Fig. 8.4. By manually measuring the time between the change in radiation and the corresponding change in outlet temperature, an estimate of the time delay at a certain flow is obtained. Repeating the procedure for different flows provides information on the effect of the flow on the time delay. In a similar way the time delay from flow to outlet temperature can be estimated by the use of such step responses. This procedure is illustrated in Fig. 8.5.

The so obtained time delays from solar radiation to outlet temperature, and from flow to outlet temperature are displayed in Tables 8.1 and 8.2, respectively. Note that there are several observations with the same flow in Table 8.1. The reason for this is that there were several dips in the radiation during a time when the flow was fixed. Due to uncertainties in the method the estimated time delay differs between the different observations.

From Tables 8.1 and 8.2 the relation between flow and time delay (in samples) was approximated with the relationship

\[ T_{D_i} = k_i Q^{-1}, \quad i = 1, 2 \tag{8.2} \]

where \( T_{D_i} \) is the time delay from solar radiation to output temperature \((i = 1)\) and from flow (and inlet temperature) to output temperature \((i = 2)\), respectively, and where \( Q \) is the flow. The constants \( k_1 = 9.13 \cdot 10^4 \, \text{samples} \cdot \text{L/h} \), and \( k_2 = 7.45 \cdot 10^4 \, \text{samples} \cdot \text{L/h} \) were determined by least squares minimization.
Figure 8.5: Time delay from flow (bottom) to outlet temperature (top). The symbols represent events that are connected.

Table 8.1: Time delay (in samples) from solar radiation to output temperature for various flows (in L/h). The sampling period $T_S = 5$ s.

<table>
<thead>
<tr>
<th>Flow [L/h]</th>
<th>Time delay [samples]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>69</td>
</tr>
<tr>
<td>1800</td>
<td>55</td>
</tr>
<tr>
<td>1800</td>
<td>50</td>
</tr>
<tr>
<td>4500</td>
<td>28</td>
</tr>
<tr>
<td>4500</td>
<td>24</td>
</tr>
<tr>
<td>4800</td>
<td>26</td>
</tr>
</tbody>
</table>

Table 8.2: Time delay (in samples) from flow to output temperature for various flows (in L/h). The sampling period $T_S = 5$ s.

<table>
<thead>
<tr>
<th>Flow [L/h]</th>
<th>Time delay [samples]</th>
</tr>
</thead>
<tbody>
<tr>
<td>562</td>
<td>140</td>
</tr>
<tr>
<td>1200</td>
<td>48</td>
</tr>
<tr>
<td>2450</td>
<td>28</td>
</tr>
<tr>
<td>2800</td>
<td>25</td>
</tr>
<tr>
<td>3150</td>
<td>24</td>
</tr>
</tbody>
</table>
Remark: As the input temperature varies very little in the setup used in this chapter, it is difficult to determine the time delay in the same way as for the radiation and the flow. This will always be the case when the collectors are operated in combination with the tanks, as the relatively large volume in the tanks will remove all major temperature variations and give the inlet temperature the behavior of a (leaky) integrator, as long as no energy is taken out of the system. However, it is not possible to manipulate the inlet temperature without risking problems associated with identification in closed loop. Since the inlet temperature does not vary much in the chosen system configuration the exact delay of the inlet temperature is unknown, but small errors in the delay are unlikely to affect the overall model performance. The delay of the inlet temperature is therefore assumed to be identical to that of the flow.

The next step is to look at the relationship between the volume of the water moving through the collectors, $V$, the time delay from the flow to the output temperature, $T_{D_2}(t)$, and the flow $Q(t)$. Conservation of mass motivate why the variables can be expected to be connected as

$$V = \int_{t-T_{D_2}(t)T_s}^{t} Q(s) ds. \quad (8.3)$$

Note that $t$ denotes discrete time here. The flow was measured at each sampling instance, whereas the volume and time delay were both unknown. To find $T_{D_2}(t)$ first note that $V$ is constant. This means that if $V$ can be determined, it will be straightforward to use numerical integration of (8.3) to determine $T_{D_2}(t)$ for all $t$, even when $Q(t)$ varies. Next, note that to find $V$ it can be used that for a constant flow $Q$, (8.3) collapses to

$$V = QT_{D_2}T_s, \quad (8.4)$$

where $T_{D_2}$ is constant. Using the relation (8.2), it follows that

$$V = QT_{D_2}T_s \approx k_2 T_s. \quad (8.5)$$

Note that $V$ is a scaled version of the volume, with the unit $L \cdot s / h$. The scaling has no practical implications, since $V$ is merely used as a constant that enables the determination of $T_{D_2}(t)$ from $Q(t)$.

When $T_{D_2}(t)$ has been determined for all values of $t$ the flow can be shifted accordingly. The shifted flow

$$Q(t) = Q(t - T_{D_2}(t)T_s) \quad (8.6)$$

will then not be delayed in time as compared to the output signal. To shift the radiation in a similar way is straightforward, since (8.2) gives that

$$T_{D_1} = k_1 Q^{-1} = \frac{k_1}{k_2} T_{D_2}. \quad (8.7)$$

When implementing the described method it is necessary to start from the final sample $t = NT_S$ to find $T_{D_1}(NT_S)$ and then iterate towards the beginning of...
the data set. In doing so there will be a section in the beginning of the measured output signal that will be unusable, in that there will be no matching input signals.

The method of estimating the time delays manually from the measured data admittedly involves approximation. There was not a perfect fit between the flow and time delays in Tables 8.1-8.2. This implies that the estimated \( k_i \) values contain some uncertainties. If the constants are overestimated the shifting will be too large, and may even make the shifted data non-causal. Hence, the computed \( k \) values were tuned to avoid such problems. Finally \( k_1 = 6.3 \cdot 10^4 \) samples \( \cdot \) L/h and \( k_2 = 6.0 \cdot 10^4 \) samples \( \cdot \) L/h were used.

The shifting also affects the transient times of the identified dynamic model. Of these, due to the transport of mass, the transient from flow to outlet temperature should show a behavior similar to that of the delays. Hence shifting should improve also the time invariance of this part of the identified model. The transient from solar radiation to outlet temperature appears intuitively to be less affected, and the modeling accuracy may therefore here be negatively affected by shifting. However, since the delays are the main concern, this is accepted in the present study. Ways to compensate for any negative impact of shifting is left for future research.

After shifting, system identification of a model without delays can be performed. To this end the data set was split in two; one subset to use for parameter estimation and one for model validation, see Fig. 8.6.

### 8.4 Identification Results

#### 8.4.1 Solar Collector Dynamics

After shifting the data to compensate for the time delays, and performing preprocessing as described above, an identification of the solar collector dynamics could be performed using the methods described in Chapters 3 and 6. As before, the underlying continuous time model structure was on the form

\[
\begin{align*}
\begin{pmatrix}
  x_1^{(1)} \\
  \vdots \\
  x_{n-1}^{(1)} \\
  x_n^{(1)}
\end{pmatrix}
&= \begin{pmatrix}
  x_2 \\
  \vdots \\
  x_n \\
  \varphi^T(x, u)\theta
\end{pmatrix}, \\
\end{align*}
\]

(8.8)

\[
y = (1 \ 0 \ \ldots \ 0)x, \\
\]

(8.9)

where \( \varphi^T(x, u)\theta \) is a polynomial in \( x \) and \( u \). To enable a comparison between different models the mean square of the prediction error,

\[
MSE = \frac{1}{N} \sum_{t=1}^{N} (y_m(t) - y(t))^2, \\
\]

(8.10)

for the simulated output of a model, evaluated for the validation data set, was used as performance measure.
Initially a linear reference model was identified. The Kalman based initialization algorithm was run with $P(0) = 0.11$, $R_{1,i} = 1.0 \cdot 10^{-2}I$, $R_{1,\theta} = 1.0 \cdot 10^{-6}I$, and $R_2 = 1.0 \cdot 10^{-2}I$ where $I$ denotes an identity matrix of appropriate order. The $R_{1,i}$ matrices are the covariance matrices for the disturbances acting on the state and parameter vectors respectively, while $R_2$ describes the covariance of the measurement noise. Further the sampling period $T_s = 5$ s was scaled by a factor $\alpha = 2$, and the initial parameters were chosen as $\hat{\theta}(0) = (0 \ldots 0)^T$. For details on the scale factor, see Chapter 3 and [96].

The final parameter vector at the end of the initialization algorithm run was used as initial values for the RPEM algorithm of Chapter 3, in accordance with [21]. The RPEM parameters used were $\Lambda(0) = 0.1$, $R(0) = 3 \cdot 10^6I$, $\mu_0 = 0.9995$, $\mu_1 = 300$, and $\mu(0) = 5$, cf. (3.18). The stability limit of the projection algorithm was chosen as $\delta = 0.01$, cf. [97].

**Step 1 - Selection of nonlinear terms**

The linear model obtained had an MSE of 1.125, which is higher than the variance of the scaled output signal $y(t)$. Clearly the linear model was insufficient to describe the solar collector dynamics, and therefore nonlinear elements were added to the model structure. To enable a large number of model structures to be tested the original algorithm tuning for the linear model was used for all the nonlinear models. The nonlinear models tested consisted of the linear model
regressor vector, with a nonlinear regressor element added to it, one at a time, to examine the effect of each parameter on the overall model performance. The added elements were polynomials up to degree 5 for \( u \), \( v_1 \), and \( v_2 \) respectively, and 1 for \( x \). The most complex element added to the linear model was consequently \( xv_1^3v_2^5u^5 \). In Fig. 8.7 the percentage of improvement compared to the linear model

\[
\Delta = \max \left( \frac{\text{MSE}_{\text{linear}} - \text{MSE}_{\text{nonlinear}}}{\text{MSE}_{\text{linear}}} , 0 \right)
\]  

(8.11)
can be seen for each of the 427 nonlinear models tried. For the cases where the MSE did not improve when a nonlinear element was added, \( \Delta \) was set to 0 since this element should clearly not be a prime candidate to be included in the final model. Note that cases where the MSE did not improve include cases where the algorithm did not converge to parameters corresponding to a stable model.

Each of the elements that improved the performance by at least 50\% (\( \Delta \geq 0.5 \)) were selected for further processing. These elements and their improvement are shown in Table 8.3.

When studying Table 8.3 there appears to be some elements that resemble each other in structure. When looking at e.g. \( v_1^4 \), \( v_1^3u \), \( v_1^3 \) and \( v_1^4u \) it appears that by increasing the polynomial degree of \( v_1 \) from 3 to 4 the MSE is reduced by 50\%. Similarly, by multiplying the terms \( v_1^3 \) and \( v_1^4 \) by \( u \), the MSE decreases.
Table 8.3: Polynomial elements that improved the $MSE$ by at least 50% when added to the linear model.

<table>
<thead>
<tr>
<th>Nonlinear element</th>
<th>$MSE$</th>
<th>$\Delta$ [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>none (linear model)</td>
<td>1.12</td>
<td>0</td>
</tr>
<tr>
<td>$v_1^2u^3$</td>
<td>0.55</td>
<td>51</td>
</tr>
<tr>
<td>$v_1^2v_2^2$</td>
<td>0.50</td>
<td>55</td>
</tr>
<tr>
<td>$v_3^3$</td>
<td>0.47</td>
<td>58</td>
</tr>
<tr>
<td>$v_3^3u$</td>
<td>0.42</td>
<td>63</td>
</tr>
<tr>
<td>$v_1^3v_2$</td>
<td>0.35</td>
<td>69</td>
</tr>
<tr>
<td>$u^3$</td>
<td>0.29</td>
<td>74</td>
</tr>
<tr>
<td>$v_4^4$</td>
<td>0.24</td>
<td>79</td>
</tr>
<tr>
<td>$v_1^4$</td>
<td>0.19</td>
<td>83</td>
</tr>
<tr>
<td>$v_1v_2^2$</td>
<td>0.15</td>
<td>87</td>
</tr>
</tbody>
</table>

by 11% and 21% respectively. One likely explanation is that the four terms describe the same phenomenon in the data, but the term $v_1^4u$ generates the best result. For this reason $v_3^3$, $v_3^3u$, and $v_4^4$ were excluded in the next step, when different combinations of promising nonlinear terms were added to the linear model structure. By the same reasoning $v_1^2v_2^2$ was excluded as it gave worse results than both $v_1^3v_2$ and $v_1v_2^2$. It can be noted that none of the best candidates included the state in the signal combination. This is a strong indication that the plant dynamics are well modeled by linear methods, but that disturbances need to be nonlinearly combined to achieve a sufficient accuracy.

Step 2 - Combination of nonlinear terms

The next step was to determine whether an even better performance could be obtained if more than one nonlinear term would be used in the model structure. The basis for this part of the experiment was to take some of the terms that each improved the performance the most, and test model structures containing different combinations of these. Remaining from Table 8.3 after removing the elements discussed above, were $u^3$, $v_1v_2^2$, $v_1^2u^3$, $v_1^2v_1$, and $v_1^4u$. None of these terms contain the state estimate, but to examine whether terms containing $x$ would improve the performance further when used in combination with the terms above, two terms containing $x$ were added to the group. These terms, $xv_1v_2^2$, and $xv_1^3v_2u$, were the elements containing $x$ that improved the model performance the most in the first step, resulting in an $MSE$ of 0.77 and 0.66 respectively.

The seven selected nonlinear elements were used in all possible combinations, along with the linear elements, as in the selection of nonlinear terms. This implies that a total of $2^7 = 128$ models (127 nonlinear and one linear) were evaluated. From Fig. 8.8 it can be seen that the best models improve the $MSE$ by between 80 and 90% as compared to the basic linear model. The
8.4. Identification Results

Figure 8.8: Improvement $\Delta$, in $\%$, for each of the models consisting of combinations of nonlinear elements and all the linear elements. For the cases where the performance was deteriorated the improvement was set to 0.

Table 8.4: Combinations of polynomial terms that improved the $MSE$ the most when added to the linear model.

<table>
<thead>
<tr>
<th>Nonlinear element</th>
<th>$MSE$</th>
<th>$\Delta$ [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>none (linear model)</td>
<td>1.12</td>
<td>0</td>
</tr>
<tr>
<td>$u^3$</td>
<td>0.29</td>
<td>74</td>
</tr>
<tr>
<td>$v_1^4u, v_1^2u^3$</td>
<td>0.25</td>
<td>78</td>
</tr>
<tr>
<td>$v_1v_2^2, v_1^2v_2$</td>
<td>0.21</td>
<td>81</td>
</tr>
<tr>
<td>$v_1^4u$</td>
<td>0.19</td>
<td>83</td>
</tr>
<tr>
<td>$v_1v_2^2, v_1^2u^3$</td>
<td>0.17</td>
<td>85</td>
</tr>
<tr>
<td>$v_1v_2^2$</td>
<td>0.15</td>
<td>87</td>
</tr>
</tbody>
</table>

combinations that resulted in the best performance can be seen in Table 8.4.

It appears that it is not possible to improve the model performance further by using combinations of the terms in Table 8.3. The model producing the best result in both experiments is the model consisting of the linear terms in combination with $v_1v_2^2$. For this model the following parameter vector was
obtained at the end of the run with the RPEM algorithm

\[
\hat{\theta}(1896T_S) = 10^{-3}(0.3885 - 1.4512 \quad 2.3344 \\
1.5612 \quad 0.0036 - 2.5511)^T.
\] (8.12)

Rescaling the parameters to compensate for the algorithm scale factor \( \alpha \), results in

\[
\hat{\theta}_{\text{rescaled}}(1896T_S) = 10^{-3}(0.7771 - 2.9025 \quad 4.6687 \\
3.1225 \quad 0.0071 - 5.1022)^T
\] (8.13)

and the matching regressor vector

\[
\varphi(x, u) = (1 \quad u(t) \quad v_2(t) \quad v_1(t) \quad v_1(t)v_2^2(t) \quad x(t))^T.
\] (8.14)

A simulation of this model over the validation data set can be seen in Fig. 8.9 along with the simulated output of the linear model. The corresponding input signals can be seen in Fig. 8.10. All signals in Fig. 8.9-8.10 have been rescaled to the magnitude of the original signals to facilitate physical interpretation of the results. The improvement of model performance associated with extending the linear model with nonlinear terms is significant. Though both the linear and
the nonlinear model show a similar behavior for the first half of the validation data set, the nonlinear model manages to describe the behavior over the second half of the data set significantly better. This is only natural as the dynamics of the solar collector can be assumed to contain nonlinear behavior.

The use of the nonlinear black-box continuous time polynomial model structure makes it straightforward to use the obtained model in a number of widely used nonlinear control schemes. In addition the good fit of the nonlinear model implies a potential use for feedforward control, supervision, and fault detection applications. Note finally that the use of the model should be limited to the flow range for which it has been validated. When using other system configurations where the maximum flow is higher the model may need further validation, or the dynamics of a second working point be investigated.

8.5 Nonlinear Feedforward MPC of the Solar Cooling Plant

In this work, pure feedforward is used to achieve the control objective. In a fully optimized commercial system, feedforward should normally be combined with feedback [83]. The feedback loop then acts to reduce the errors that remain after feedforward compensation. The so reduced error allows for an increase
of the margins of the feedback controller, or for an increase of the bandwidth. The separated study of the feedforward part performed in the present chapter is believed to be a necessary first step since it is only without feedback that the impact of the accuracy of the identified nonlinear model can be studied in detail. This study alone is quite complicated due to the time varying, flow dependent long delays in combination with the nonlinear effects caused by the physics of the heat transfer and solar radiation.

8.5.1 Controller Overview

The model obtained in the previous section will in this section be used for model based control. Towards that end, first remember that the model was identified using delay compensated, shifted data. This means that the time delay is not considered in the model, and must consequently be handled otherwise. Another thing worth noting is that due to the time delays, the effect of changes in disturbances or the input signal at time \( t \) will not show up in the output temperature until time \( t + T_{D_1}(u)T_S \) or \( t + T_u(u)T_S \) respectively. This means that the model based control will have to be based on predicted values of the states. Note that \( T_u = T_{D_2} \) will be used to denote the input signal delay in the discussion on control that follows.

The controller uses the previously obtained model, together with the measured disturbances \( v_1(t), v_2(t) \) and the reference signal \( y_{ref}(t) \) to calculate the optimal control. The control is then fed to the system. A schematic picture of the system/controller can be seen in Fig. 8.11.

8.5.2 Flow Dependent Sampling

To compensate for the flow dependent time delays the controller utilizes a sampling period different from the regular equidistant time sampling used by the computer running the system and the controller. This internal sampling period in the controller depends on the flow. The idea is that by sampling at a rate that is proportional to the inverse of the flow the time delay will become
constant as measured in terms of the number of samples inside the controller. The values of each signal at the flow dependent sampling instances can easily be obtained by interpolation of the measured signals, see (7.20). It is thereafter straightforward to use the model obtained from system identification earlier. The calculated control signal can be transformed back to the original sampling instances through interpolation, as described in (7.21). As stated earlier there is an additional effect on transients, this is however neglected in the present study. The details of the flow dependent sampling algorithm appear in Chapter 7.

### 8.5.3 Extended Integrating Plant Model

The first order continuous time model of the solar collector dynamics that was identified in Section 8.3 was on the form

\[
\dot{x}(t) = \varphi^T(t, u(t - T_{D_1} T_{\bar{s}}), v_1(t - T_{D_1} T_{\bar{s}}), v_2(t - T_{D_2} T_{\bar{s}}))\theta
\]

(8.15)

\[
\dot{y}(t) = x(t)
\]

(8.16)

with the identified parameter vector

\[
\theta = 10^{-3}(0.7771 \quad -2.9025 \quad 4.6687 \quad 3.1225 \quad 0.0071 \quad -5.1022)^T.
\]

(8.17)

Here \(T_{\bar{s}}\) is the internal (flow dependent) sampling period of the controller. The delays have been obtained using the shifting delay model of (8.2). The so obtained delays have been re-introduced after identification using shifted data. The flow dependent sampling is run to maintain these delays in the core of the controller. The use of \(\dot{y}(t)\) is introduced to stress the fact that the scope of the controller design is model based (predictive) feedforward control, hence no measured output signals are used directly in the controller. With such a model structure it is straightforward to implement e.g. quadratic criteria where the input is penalized along with the error in the state or output. An example of such a criterion could be

\[
\int_{s=0}^{M^+} ((y_{r e f}(t + sT_{\bar{s}}) - \dot{y}(t + sT_{\bar{s}}))^2 + \rho u^2(t + sT_{\bar{s}} - T_u T_{\bar{s}}))ds.
\]

(8.18)

where \(y_{r e f}(t)\) denotes the reference signal and \(M^+\) constitute the continuous time control horizon (cf. Section 7.4). The handling of the non-standard delay \(T_u\) was discussed at length in Chapter 7. Hence, for the present discussion the formulation with a delay does not introduce any problems.

For the solar collector control problem, this choice of criterion with a penalty on the input signal would cause problems with the controller. First, a controller based on (8.18) will cause the input signal to be as small as possible as long as the error does not get too large. An unnecessarily small control signal (flow) causes the delays to increase and hence the system may become more difficult to control. It would be reasonable to assume that the criterion (8.18) could give an oscillative control signal. This is neither beneficial from a control point of view, nor from a pump maintenance perspective. Secondly, and more importantly, since the signals are not zero mean a penalty on the input would
result in a significant steady state bias error since control penalty is balanced against control error in (8.18).

To solve the above two issues it is instead preferable to penalize the derivative of the input, \( i.e. \) to make the outlet temperature follow the reference value closely while not changing the flow too rapidly. The criterion would then be formulated as

\[
\int_{s=0}^{M^+} \left((y_{\text{ref}}(t + sT_{\bar{s}}) - \hat{y}(t + sT_{\bar{s}}))^2 + \rho \hat{u}^2(t + sT_{\bar{s}} - T_uT_{\bar{s}})\right)ds. \quad (8.19)
\]

To incorporate the model structure (8.15)-(8.16) into an optimization algorithm with a criterion function like (8.19), the model needs to be rewritten. In order to do so, the state vector is extended by the delayed input,

\[
x_E(t) = (x(t) \quad u(t - T_uT_{\bar{s}}))^T, \quad (8.20)
\]

and the model is written as

\[
\dot{x}_E(t) = \left( \varphi^T(t, x_E(t), v_1(t - T_D, T_{\bar{s}}), v_2(t - T_D, T_{\bar{s}})) \theta \right) \\
+ \left( \begin{array}{c} 0 \\ 1 \end{array} \right) \hat{u}(t - T_uT_{\bar{s}}) \quad (8.21)
\]

\[
\hat{y}(t) = (1 \quad 0)x_E(t) \quad (8.22)
\]

As the input is now included in the state vector, and a new input, \( \hat{u} \), corresponding to the derivative of the input is introduced, (8.15) and (8.18) can be replaced by (8.21) and (8.22). The criterion (8.19) is then transformed into

\[
\int_{s=0}^{M^+} \left((y_{\text{ref}}(t + sT_{\bar{s}}) - \hat{y}(t + sT_{\bar{s}}))^2 + \rho \hat{u}^2(t + sT_{\bar{s}} - T_uT_{\bar{s}})\right)ds. \quad (8.23)
\]

Note that this is the counterpart to the common discrete time linear strategy to use the differentiated model equations for the controller design, thereby introducing integrator action in the controller. This follows since the applied control needs to be integrated from \( \hat{u}(t - T_uT_{\bar{s}}) \) as defined by the control algorithm (7.19). Hence, the applied control will be

\[
u(t) = u(t - T_{\bar{s}}) + T_{\bar{s}}\hat{u}(t - T_{\bar{s}}). \quad (8.24)
\]

In the rest of the chapter a discretized formulation is used to be able to reuse the results of Chapter 7. To obtain a discrete time equivalent useful in the coming sections, the model (8.21), (8.22) is discretized using a forward Euler equation, while the integral of the criterion is exchanged by a summation. After division by \( T_{\bar{s}} \) this gives the discrete time criterion

\[
J = \sum_{s=0}^{N-1} \left(y_{\text{ref}}(t + sT_{\bar{s}}) - (1 \quad 0)x_E(t + sT_{\bar{s}}))^2 + \rho \hat{u}^2(t + sT_{\bar{s}} - T_uT_{\bar{s}})\right). \quad (8.25)
\]
subject to

\[
x_E(t + T_{\bar{s}}) = x_E(t) + T_{\bar{s}} \left( \varphi_x^T(t, x_E(t), v_1(t - T_{D_1}T_{\bar{s}}), v_2(t - T_{D_2}T_{\bar{s}})) \theta \right) \\
+ T_{\bar{s}} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \hat{u}(t - T_u T_{\bar{s}})
\]

Equations (8.25)–(8.27) define the optimization problem used for feedforward control in the rest of the chapter. The solution of this type of optimization problem requires a solution of the Euler–Lagrange equations, see [25]. These equations were discussed in Chapter 7.

### 8.6 Feedforward Control Results

The solar radiation is a quantity that can be measured but which cannot be affected by the controller. Under certain circumstances, like when the weather is partly cloudy, the radiation varies a lot, and sometimes with a relatively high frequency. This results in large variations in the outlet temperature from the solar collectors. This in turn complicates the use of the solar collectors in combination with the absorption machine, which requires high and even inlet temperatures to operate efficiently [48].

This section focuses on the results of experiments where the objective is to keep the outlet temperature as closely as possible to a reference temperature, using the feedforward controller described above. The plant has been used to collect the measurements of solar radiation and inlet temperature that are used to obtain the results of this section. However, so far it has not been possible to test the feedforward regulator in Seville, although all software has been developed. The reason for this is that a successful integration requires quite a long time spent on-site. Since it is not yet clear whether the needed experiment time and funding will be available, simulated results using real measured disturbances are presented.

The same model structure is used for simulations of the system and for controller design in the first experiments reported here. The model used for these purposes is the nonlinear model (8.26), (8.27), together with (8.13), (8.14) obtained from the identification experiments of section 8.3. This means that the behavior of the controller can be validated without the influence of unmodeled dynamics. The criterion was given by (8.25) subject to

\[
x_E(t + T_{\bar{s}}) = x_E(t) + T_{\bar{s}} \left( \varphi_x^T(t, x_E(t), v_1(t - T_{D_1}T_{\bar{s}}), v_2(t - T_{D_2}T_{\bar{s}})) \theta \right) \\
+ T_{\bar{s}} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \hat{u}(t - T_u T_{\bar{s}})
\]

\[
\hat{y}(t) = (1 \ 0)x_E(t)
\]
Figure 8.12: Feedforward control performance, using measured disturbances and the identified model. The thin solid line corresponds to the output reference signal, the thick solid line is the simulation with control, while the dashed lines correspond to simulations with constant flow (i.e. without control).

where

\[ f(x_E(t), v_1(t - T_{D1}T_{\bar{s}}), v_2(t - T_{D2}T_{\bar{s}})) = 10^{-3}(0.7771 - 2.9025x_{E,2}(t) + 4.6687v_2(t - T_{D2}T_{\bar{s}}) + 3.1225v_1(t - T_{D1}T_{\bar{s}}) + 0.0071v_1(t - T_{D1}T_{\bar{s}})v_2^2(t - T_{D1}T_{\bar{s}}) - 5.1022x_{E,1}(t)). \]  

(8.30)

Note that \(x_{E,1}(t)\) and \(x_{E,2}(t)\) denote the two states of \(x_E(t)\) (i.e. \(x(t)\) and \(u(t - T_0T_{\bar{s}})\) respectively).

The reference signal was chosen to include a ramp as well as a step change. The reference is shown as a thin solid line in Fig. 8.12. The thicker solid line of Fig. 8.12 represents the simulation with a controller where \(k = 10^{-4}\), \(\varepsilon = 10^{-1}\) and \(\rho = 100\), cf. Section 7.1. For comparison Fig. 8.12 also contains two simulations with constant inputs, i.e. without control, for two different flows. The flows were each kept at 2500 and 5000 \(L/h\) respectively, and though these are not the limits of the controller operating range they still provide information of how the system would behave if no controller was used. The curve with the higher temperature refers to the case with lower flow, as can be expected. The corresponding control signals of the experiments with (solid) and without (dashed) control respectively can be found in Fig. 8.13. From Fig. 8.12 it is also clear that the use of the proposed controller significantly
improves the system behavior compared to the cases with constant flow through the collectors. Though this is hardly surprising, it is clear that the ability to reduce the flow cuts off the dips in the outlet temperature that occur after periods of low radiation. Similarly the ability to keep the temperature down when the radiation is high depends on the controller being able to increase the flow. It is also evident from Fig. 8.12 that the effect of the control is subject to limitations, and that the controller cannot fully compensate for all variations in radiation. This is, however, not a controller problem but a system design issue. From the two curves corresponding to constant flows it can be seen that the control range is too small, and as the radiation drops it is often not even theoretically possible to keep the temperature at a fixed setpoint. This is particularly obvious as the controller keeps hitting the upper and lower bounds of the flow. At a number of sections of the data, e.g. towards the end of the run, the controller cannot bring the temperature down to the reference, even though the flow is at its maximum. The only way to avoid that the controller reaches the flow boundaries, is to adjust the reference signal gradually as the inlet temperature increases. Note that the upper and lower levels of the control signal have not been incorporated in the controller, but have been added as a saturation function of the control signal before feeding it to the plant. The reason is a desire to highlight feedforward performance.
The second set of experiments illustrate the effect of modeling errors. For this purpose simulations were run using randomized 5% differences between the system and model parameters. The parameter vector was randomized as

\[ \theta_m = (1 + 0.05 \times 2 \times (\text{rand}(\text{size}(\theta_s)) - 0.5)) \times \theta_s \]  

(8.31)

Here \( \theta_s \) are the system parameters, \( \theta_m \) are the model parameters, and \( \text{rand()} \) generates a uniform distribution on \([0,1]\). The result is shown in Fig. 8.14 and Fig. 8.15. The thick solid line is the behavior without modeling errors (as in Fig. 8.12) and the two dashed curves represent the two worst cases of 10 different model error realizations. No additional problems e.g. with stability occurs. Since feedforward control is used, the static behavior deviate somewhat from the nominal model.

### 8.7 Summary

The chapter has discussed nonlinear feedforward MPC control of the outlet temperature of a solar collector pilot plant at the University of Seville, Spain. The control challenges include plant nonlinearities, long time delays which depend

---

**Figure 8.14:** Randomized feedforward control performance, using measured disturbances and the randomized identified model. The thin solid line corresponds to the output reference signal, the thick solid line is the simulation without model errors, while the dashed lines correspond to two examples of model errors with a high impact on controller performance.
Figure 8.15: Feedforward control signal, using measured disturbances and the randomized dynamic models. The solid line represents the simulation without model errors, while the dashed lines correspond to two examples of model errors with a high impact on controller performance.

strongly on the controlling flow of water, as well as large disturbances in terms of varying solar radiation and inlet temperature. A key in the development was the application of flow dependent shifting. This allows identification of a continuous time nonlinear model with constant delays using measured inputs, outputs and disturbances. After identifying and evaluation 428+127 nonlinear polynomial model structures, it was found that the model with the least mean square error was linear in the state and input signal. The disturbances however needed to be nonlinearly combined. The so obtained model outperformed the best linear model. From a control perspective, this is highly interesting since it suggest a combination of nonlinear feedforward control and linear feedback.

An extended nonlinear model, including an integrator, was then defined from the identified model. This extended model was used for feedforward control in an MPC setting where the criterion was minimized using iterative prediction of the output, a necessity due to the long delays. The variation of the delays were removed by application of flow dependent sampling, encapsulating the control algorithm in a constant delay environment. The controller was evaluated by simulation, using measured disturbances from the plant. Although the control signal range was found to be insufficient, the controller was able to significantly reduce the disturbance impact on the outlet temperature, using only feedforward. The software used for identification [99] and feedforward control [18] are available for free download.
Conclusions

9.1 Contributions and Results

Much of the work in this thesis has its starting point in the nonlinear identification algorithm in Chapter 3. In the anaerobic digestion and solar heating examples of Chapter 4 this algorithm was used to model nonlinear system dynamics. The anaerobic digestion example illustrates how a complex nonlinear system can sometimes be described by relatively simple nonlinear black-box models. The result has implications for controller design for the particular system. It was also discussed how tuning and choice of initial parameters affect the algorithm performance. In the domestic solar heating example the data set was small. For a recursive identification method this could pose a problem, as the initial parameters may lie far from the optimal parameters, and tuning to increase convergence speed can cause instability. The problem was solved by the use of multiple scans, and the final model showed similar behavior to a previously developed grey-box model.

In Chapter 5 the convergence properties of the RPEM algorithm in Chapter 3 were analyzed. In doing so a number of conditions on the algorithm and the data were formulated. It was proven that provided that these conditions hold the algorithm will converge either to a minimum of the criterion function, or to the boundary of the model set.

For nonlinear identification problems with non-convex criterion functions, e.g. output error identification methods, there is a risk of convergence to a suboptimal minimum of the criterion function. The examples in Chapter 4 and the convergence analysis in Chapter 5 show that the choice of initial parameters for the RPEM algorithm in Chapter 3 is of great importance for the final result. Preferably the initial parameters should be close to the parameter vector corresponding to the best description of the system. This issue was addressed in Chapter 6, where an algorithm based on Kalman filter theory was proposed for providing initial parameters for the RPEM. The proposed algorithm is based on an approximation using differentiated data, thereby avoiding convergence
to suboptimal minimum points. For this reason the algorithm is more sensitive to noise than the RPEM. Several examples were used to compare algorithm performance and to show how the method can be successfully used for its intended purpose. A software implementation of the algorithm, available for download, was also presented.

In Chapter 7 model predictive control of systems with long input dependent time delays were discussed. In the present thesis the approach has been to embed a controller for constant time delays in a framework of input dependent sampling, and to apply the controller in an MPC setting. Chapter 7 addresses issues related to this topic by discussing how computational complexity can be kept down by a good choice of criterion function for an optimal control problem with long and constant time delays, and by illustration of how input dependent sampling can be used to convert a control problem with input dependent time delays to a problem with constant time delay. The results were used in development of a software package for model predictive control of systems with input dependent time delays.

The system identification and control methods discussed in Chapter 3, 6, and 7 were used in Chapter 8 to model and control the outlet temperature of a solar collector field. The system has nonlinear dynamic behavior and long, input dependent time delays. In order to perform the identification experiments, the data was shifted to compensate for the varying time delays. 428+127 different nonlinear model structures were evaluated, and the best model had nonlinear terms in the disturbances only. The model was used in a feedforward model predictive controller, where the controller for constant time delays was embedded in an input dependent resampling framework, as suggested in Chapter 7. The resulting controller was compared to the case with constant input (no control). The effect of modeling errors in the controller was also studied.

The initialization method proposed in Chapter 6 has made it significantly easier to use the RPEM algorithm for system identification experiments. All in all, the method has in this thesis been used for identification of two simple numerical examples and two physical systems (the tank process in Chapter 6 and the solar collector dynamics in Chapter 8) from which live data has been collected. The results show that the Kalman filter based algorithm can successfully be used to provide initial parameters for more advanced nonlinear search algorithms like the RPEM of Chapter 3. As the regressor vector of the initialization algorithm is generated by differentiation of the measured output, the use of the method is believed to be more difficult to use for systems of high order, and where the measurements are noisy. However, there are still many systems where the proposed algorithm can be expected to work well.

The large and varying time delays of the solar collectors in Chapter 8 makes the control problem interesting and challenging. It is far from obvious how to address the problem, and as shown in the survey of [28; 29], lots of approaches have been used in the past, many of which work well, at least under certain conditions. It should be pointed out that the modeling of the solar collectors in Chapter 8 has been performed using data from a day with partly cloudy weather, which implies large and rapid changes in solar radiation. The controller has also been developed using the model obtained from this data. Con-
sequently it has been developed to handle this kind of weather, which makes the control problem significantly more difficult than it is on a sunny summer day with clear sky. Meanwhile, it is obvious from the experiments that one of the greatest problems when trying to control the solar collector outlet temperature is the narrow control range of the system. The controller manages to compensate for a significant portion of the temperature variations that would otherwise have been the consequence of the radiation variations. However, it is highly unlikely that another controller could have cancelled out the remaining part of the variations. The reason is simply that the controller uses much of the control range already, and it does not seem to be enough. It is possible that the performance of the present controller could be improved somewhat by a different choice of lower limit of the allowed flows. This effect could increase the temperature at times when the radiation is too low to bring the outlet temperature up to the reference. However, the problem of having the control signal reaching the maximum flow without being able to bring the outlet temperature down to the reference can only be compensated for by a change in the reference signal, which seems rather backwards from a control point of view. The relevant way forward hence seems to be a redesign of the plant. This points to a final important contribution, namely the usefulness of studies as the one of Chapter 8, in support of the initial design of future solar cooling plants.

9.2 Future Research

In the course of research the answering of one question often raises new questions. Some problems of interest for future research that have occurred during the work that led up to this thesis are listed next.

- A natural continuation of the anaerobic digestion example in Chapter 4 would be to study the model behavior when applied to measured data from a biogas reactor, to design nonlinear controllers based on the model, and eventually to evaluate the controller performance on a real reactor. The model of the solar heating system in the same chapter could also be used for controller design, as was done for the solar collectors of the solar cooling system in Chapter 8 of this thesis. The model could also potentially be used for monitoring and fault detection.

- Considering the results of Chapter 5 it might be of interest to study other fundamental properties of the RPEM algorithm with theoretical analysis to get a more complete view of the algorithm and how it performs, and compare it to other methods. Examples include tracking properties and asymptotic accuracy.

- As pointed out, it can be assumed that the differentiation of the measured output in the regressor generation makes it difficult to use the initialization algorithm of Chapter 6 successfully for high order systems/models and very noisy data. Future studies of the sensitivity to high model orders and measurement noise could be performed to verify the assumption and evaluate the sensitivity. It would also be of interest to compare the
performance with other nonlinear identification methods, e.g. utilizing NOE model structures.

- The work on the controllers presented in Chapter 7 has shown that due to the finite horizon of the MPC control problem, some static errors of the output may remain even with integral action. Further studies on this subject is required to evaluate the extent of the problem and possibly suggest control strategies to counteract offset problems in MPC.

- A number of future research topics related to the solar collector control problem in Chapter 8 include
  - Addition of feedback to the controller.
  - Inclusion of upper and lower limits of the flow in the optimization criterion.
  - Testing of the controller on the plant (open and closed loop) to evaluate the performance on the real system. Note that this requires a significant amount of on-site time.
  - Studies how the resampling affects the time constants of the system. It is likely that the approximation can be more motivated from a physical point of view for the flow to outlet temperature dynamics than for the radiation to outlet temperature case. Perhaps a re-transformation of the radiation signal after compensating for the time delays would result in an even better model of the system dynamics.

This concludes the thesis. Hopefully the methods and results discussed herein will be of use for handling of other identification and control problems, as well as constituting a contribution to further development of solar heating and cooling technology. Refining the techniques for utilization of sustainable energy sources will make these forms of energy more competitive and thereby a more attractive alternative to the currently dominating fossil fuels.
Part III

Appendices
Proof of Theorem 1 (Chapter 5)

A.1 Proof of Lemma 1 - Reformulation of (5.4)

In order to be able to write the algorithm (5.4) in the form of (5.5), introduce the vector

$$\xi(t) = \left( x^T(t) \left( \text{col} \left( \frac{dx}{d\theta}(t) \right) \right)^T x_{ym}(t) x_u^T(t) x_w^T(t) \tilde{u}^T(t) \tilde{w}^T(t) \right)^T.$$  (A.1)

Here $x_{ym}(t)$ is the state vector of the nonlinear difference equation generating the undisturbed part of $y_m(t)$, cf. S2. Similarly $x_u(t)$ is the state vector of the nonlinear difference equation generating the input signal, cf. M2. Finally $x_w(t)$ is the state vector of the bounded linear difference equation generating the colored disturbance vector $w(t)$ from $\tilde{w}(t)$. Note that boundedness of the system can be secured by the saturation technique applied to the model of the algorithm. The notation $\text{col}(\cdot)$ stacks the columns of a matrix on top of each other to form a vector.

The idea is now that the pseudo data vector $\tilde{z}(t) = (\tilde{u}^T(t) \tilde{w}^T(t))^T$ fits into the notation of (5.5) with $e(t) = \tilde{z}(t)$. This is the reason why stochastic modeling is introduced also for $y_m(t)$ and $u(t)$. The generation of $u(t)$ and $y_m(t)$ then needs to be analyzed. Using S2, it follows that

$$x_w(t) = \text{sat}_a(Fx_w(t-1) + \tilde{w}(t)) \triangleq \kappa_w(\xi(t-1), \tilde{z}(t))$$  (A.2)

where $F$ is a Hurwitz matrix and where $\kappa_w(\cdot, \cdot)$ is a continuously differentiable function. Then, using (A.1), M2, and S2 it follows that

$$x_u(t) \triangleq \kappa_u(x_u(t-1), \tilde{u}(t)) = \kappa_u(\xi(t-1), \tilde{z}(t))$$  (A.3)

where the vector function $\kappa_u$ defines the nonlinear difference equation of M2. Furthermore, (A.1), M2, S2, and (A.3) results in

$$x_{ym}(t) = \bar{k}_{ym}(x_{ym}(t-1), u(t)) = \bar{k}_{ym}(x_{ym}(t-1), \kappa_u(\xi(t-1), \tilde{z}(t))$$

$$\triangleq \bar{k}_{ym}(\xi(t-1), \tilde{z}(t)).$$  (A.4)
Using (A.2)-(A.4) and S2, $\varepsilon(t)$ and $\psi(t)$ can be rewritten as

$$
\varepsilon(t) = y_m(t) - y(t) = y_m(t) - C_m x(t) = (-C_m 0 C_{y_m} 0 C_w 0 0) \xi(t), \triangleq \zeta_\varepsilon(\xi(t)),
$$

(A.5)

$$
\psi(t) = C_m \frac{dx}{d\theta} = C_m^{col} \frac{dx}{d\theta} = (0 C_m^{col} 0 0 0 0) \xi(t) \triangleq \zeta_\psi(\xi(t))
$$

(A.6)

where $C_m^{col}$ is a matrix composed of the constant components of $C_m$. From (5.4), $\varphi(t)$ is a function of $x(t)$ and $u(t)$, where $u(t)$ is by (A.3) a function of $\xi(t - 1)$ and $\bar{z}(t)$. $x(t)$ is a function of $x(t-1)$, $\varphi(t-1)$ and the first component of $\theta^*(t-1)$, where

$$
\theta^* = (\theta^T (\text{col} R)^T (\text{col} \Lambda)^T)^T
$$

(A.7)

is the extended parameter vector. Hence, $\varphi(t)$ depends on $\theta^*(t-1)$, $\xi(t-1)$, and $\bar{z}(t)$. It follows that $\varphi(t)$ and $\frac{d\varphi}{dx}(t)$ can be written as

$$
\varphi(t) \triangleq \zeta_\varphi(\theta^*(t-1), \xi(t-1), \bar{z}(t))
$$

(A.8)

$$
\frac{d\varphi}{dx}(t) \triangleq \zeta_{\frac{d\varphi}{dx}}(\theta^*(t-1), \xi(t-1), \bar{z}(t)).
$$

(A.9)

By the same reasoning, if $\zeta_\varphi(\cdot)$ and $\zeta_{\frac{d\varphi}{dx}}(\cdot)$ are inserted into the expressions for $x(t)$ and $\frac{dx}{d\theta}(t)$ of (5.4) it is clear that

$$
x(t) \triangleq \eta_x(\theta^*(t-1), \xi(t-1), \bar{z}(t))
$$

(A.10)

$$
\frac{dx}{d\theta}(t) \triangleq \eta_{\frac{dx}{d\theta}}(\theta^*(t-1), \xi(t-1), \bar{z}(t))
$$

(A.11)

The algorithm can now be written in the form of (5.5) by the use of the definitions of Lemma 1, and (A.1)-(A.11).

The open set $D_R$ of [54] and [53] corresponds to $D_M \setminus \partial D_M$ of this paper, while the closed set $\bar{D} \subset D_R$ of [53] corresponds to the compact set $\bar{D} \subset D_M \setminus \partial D_M$.

### A.2 Verification of Regularity Conditions

#### Verification of C1

Because of the saturation imposed on the right hand-side of (5.4) on $x$ and $\frac{dx}{d\theta}$, and since M2 and S2 ensures that $x_{y_m}$, $x_u$, $x_w$ $\bar{u}(t)$ and $\bar{w}(t)$ are bounded, all components of $g(x_g, \varphi_g, \varepsilon)$ are bounded. C1 follows.

#### Verification of C2

Continuous differentiability of $Q(t; x_g, \varphi_g)$ follows from M1, M2, S2, (5.4), (5.6), (A.5), and (A.6), by inspection. The derivatives with respect to $x_g$ and $\varphi_g$ shall also be bounded in $t$ for $x_g \in D_R(= (D_M \setminus \partial D_M))$. To study boundedness the
partial derivatives of $Q(t; x_g, \varphi_g)$ are studied in a neighborhood of $x_g = \bar{x}_g$ and $\varphi_g = \bar{\varphi}_g$. It holds that

$$
\left\| \frac{\partial Q}{\partial x_g} \right\| \leq \sum_{i} \sum_{j} \left\| \frac{\partial Q_i}{\partial x_{g,j}} \right\|, \quad \left\| \frac{\partial Q}{\partial \varphi_g} \right\| \leq \sum_{i} \left\| \frac{\partial Q_i}{\partial \varphi_g} \right\|. \quad (A.12)
$$

To illustrate the procedure similar to that of [58], consider $\frac{\partial Q_1}{\partial \varphi_g}$.

$$
\left\| \frac{\partial Q_1}{\partial \varphi_g} \right\| = \left\| \mu(t) \frac{\partial}{\partial \xi} (R^{-1} \zeta(\xi) \Lambda^{-1} \zeta(\xi)) \text{gate}(\Gamma(\theta)) \right\|
= \mu(t) R^{-1} \left( \frac{\partial \zeta(\xi)}{\partial \xi} \Lambda^{-1} \zeta(\xi) + \zeta(\xi) \Lambda^{-1} \frac{\partial \zeta(\xi)}{\partial \xi} \right)
\times \text{gate}(\Gamma(\theta)) \leq \mu \delta_{\Lambda}^{-1} \delta_{R}^{-1} \left( \left\| \frac{\partial \zeta(\xi)}{\partial \xi} \right\| \left\| \zeta(\xi) \right\| + \left\| \zeta(\xi) \right\| \left\| \frac{\partial \zeta(\xi)}{\partial \xi} \right\| \right) \leq C \mu \delta_{\Lambda}^{-1} \delta_{R}^{-1} (1 + \|\xi\|). \quad (A.13)
$$

The second last inequality follows from M1 and G1 by noting that the projection algorithm is inactive for sufficiently large times, cf. [92], Appendix A. The last equality follows from (A.5) and (A.6). Now, the model and the system are exponentially stable and bounded (M1, M2 and S2), provided that $x_g \in D_R$. Hence, in the considered neighborhood

$$
\|\xi\| \leq \|\xi - \bar{\xi}\| + \|\bar{\xi}\| \leq C(1 + \|\bar{\xi}\|) \leq C \Rightarrow \left\| \frac{\partial Q_1}{\partial \xi} \right\| \leq C \quad (A.14)
$$

where the constant $C$ is formed from $\mu$, $\delta_{R}^{-1}$, $\delta_{\Lambda}^{-1}$ and the constants from the regularity conditions. In the same way it can be shown that the bound is valid for all partial derivatives of (A.12), i.e.

$$
\left\| \frac{\partial Q}{\partial x_g} \right\| < C, \quad \left\| \frac{\partial Q}{\partial \varphi_g} \right\| < C. \quad (A.15)
$$

**Verification of C3**

Follows from M1, M2, S2 and (5.4) by inspection.
Verification of C4

Iteration of (5.4) gives
\[ \varphi_g(t) - \tilde{\varphi}_g(t) = \frac{\partial g}{\partial \varphi_g}(\tilde{\varphi}_g(t - 1))(\varphi_g(t - 1) - \tilde{\varphi}_g(t - 1)) \]
\[ + \frac{\partial g}{\partial x_g}(\tilde{x}_g(t - 1))(x_g(t - 1) - \tilde{x}_g) = \]
\[ \frac{\partial g}{\partial x_g}(\tilde{x}_g(t - 1))(x_g(t - 1) - \tilde{x}_g) + \sum_{i=2}^{t-n} \left( \frac{\partial g}{\partial x_g}(\tilde{x}_g(t - i)) \right) \times (x_g(t - i) - \tilde{x}_g) \frac{1}{j=1}^{i-1} \frac{\partial g}{\partial \varphi_g}(\tilde{\varphi}_g(t - j)) \].
\[ (A.16) \]

The first equality follows by application of the mean value theorem (\( \tilde{\varphi}_g(t - 1) \) is a point between \( \varphi_g(t - 1) \) and \( \tilde{\varphi}_g(t - 1) \)). This is valid since \( g \) is smooth by M1, M2 and S2, and (A.16) then follows by repeated application of this argument. The projection algorithm and M1, M2 and S2 now guarantees that (cf. (5.4), (5.3) and (A.10))
\[ \| \frac{\partial g}{\partial \varphi_g}(t) \| \leq 1 - \delta < 1, \forall t. \]  
(A.17)

Hence,
\[ \| \varphi_g(t) - \tilde{\varphi}_g(t) \| \leq \| \frac{\partial g}{\partial x_g}(\tilde{x}_g(t - 1))\| \| x_g(t - 1) - \tilde{x}_g \| \]
\[ + \sum_{i=2}^{t-n} \left( \prod_{j=1}^{i-1} \| \frac{\partial g}{\partial \varphi_g}(\tilde{\varphi}_g(t - j))\| \right) \| \frac{\partial g}{\partial x_g}(\tilde{x}_g(t - i))\| \times \| (x_g(t - i) - \tilde{x}_g) \| \]
\[ \leq \sum_{i=1}^{t-n} (1 - \delta)^{i-1} \max_{n \leq s \leq t} \| \frac{\partial g}{\partial x_g}(s)\| \max_{n \leq s \leq t} \| x_g(s) - \tilde{x}_g \| \]
\[ < \frac{1}{\delta} C \max_{n \leq s \leq t} \| x_g(s) - \tilde{x}_g \| < C \max_{n \leq s \leq t} \| x_g(n) - \tilde{x}_g \|. \]
(A.18)

Here continuous differentiability (C3) and the conditions M1, M2, S2 (boundedness of signals) was used to bound \( \max_{n \leq s \leq t} \| \frac{\partial g}{\partial x_g}(s)\| \).

Verification of C5

Condition C5 follows by repeated use of the exponential stability of \( x \) and \( \frac{dx}{dt}, x_u, x_w \) (M1 and S2) together with S1. The details are similar to Appendix 4A of [58]. They are not repeated here.

Verification of C6

The verification of C6 follows [92] in all details, with the difference that \( \Lambda \) is added here. For completeness, the full argument is repeated here.
Note that A2 consists of unprojected average updating directions, whereas (5.6) is affected by the gate function. Since the signals are bounded by M1, this is of no consequence, since the unprojected updating direction coincide with the projected one for interior points of $D_M$. To demonstrate this the $\theta$-recursion of (5.4) is considered. For fixed $(\theta, R, \Lambda) \in D_M \setminus \partial D_M$, M1, M2, A1 and S2 show that $R^{-1}\psi(t)\Lambda^{-1}\epsilon(t)$ is bounded. Consequently, using G1, $rac{\mu(t)}{t}R^{-1}\psi(t)\Lambda^{-1}\epsilon(t) \to 0$ as $t \to \infty$. Following (5.7) $\Gamma_{\theta}(\cdot) = |\lambda_1(S(\theta))|$ is chosen, where $\lambda_1(S(\theta))$ is the eigenvalue of the linearized system $S(\theta)$ with the largest absolute value. $D_M$ is restricted by only allowing eigenvalues of $S(\cdot)$ with absolute values in $(0, 1-\delta)$ for some small $\delta > 0$. By using $I_0 = (0, 1-\delta)$, (5.4), (5.5), (5.7), (A.16), and A2 the projected direction can be written as

$$\lim_{t \to \infty} E\mu(t)R^{-1}\psi(t)\Lambda^{-1}\epsilon(t)$$

$$\times \text{gate}(|\lambda_1(S(\theta(t-1) + \frac{\mu(t)}{t}R^{-1}\psi(t)\Lambda^{-1}\epsilon(t))|))$$

$$= \mu R^{-1} \lim_{t \to \infty} E\psi(t)\Lambda^{-1}\epsilon(t)\text{gate}(|\lambda_1(S(\theta))|)$$

$$= \mu R^{-1} \lim_{t \to \infty} E\psi(t)\Lambda^{-1}\epsilon(t) = \mu R^{-1} f(\theta)$$

which is the unprojected direction. The last equality follows from A2. A similar approach is used in [92], p. 2194.

**Verification of C7**

Follows since the components of $e(t) = \tilde{z}(t)$ are i.i.d.

**Verification of C8-C11**

C8-C11 follows immediately for $\gamma(t) = \frac{1}{t}$.

### A.3  Verification of the Boundedness Condition

The boundedness condition of Theorem 1 of [53] assumes the existence of a $C < \infty$ and an infinite subsequence $\{t_k\}$ such that $(\theta(t_k), R(t_k), \Lambda(t_k)) \in D_M$ and $\|\xi(t_k)\| < C < \infty$. The projection algorithm (5.3) ensures that $(\theta, R, \Lambda) \in D_M$ is compact there is at least one cluster point $(\tilde{\theta}, \tilde{R}, \tilde{\Lambda}) \in D_M$ (Bolzano-Weierstrass). The case with cluster points on $\partial D_M$ is covered by Lemma 3. As for the case of cluster points $(\tilde{\theta}, \tilde{R}, \tilde{\Lambda}) \in D_M \setminus \partial D_M$, standard analysis shows that since $D_M \setminus \partial D_M$ is open, there is a compact set $\overline{D}_M$ and a neighborhood $B(\tilde{\theta}, \tilde{R}, \tilde{\Lambda}, \Delta) = \{(\theta, R, \Lambda) \mid (\theta, R, \Lambda) \in B(\tilde{\theta}, \tilde{R}, \tilde{\Lambda}, \Delta) \subset \overline{D}_M \subset D_M \setminus \partial D_M$, where $\Delta > 0$. Since $(\tilde{\theta}, \tilde{R}, \tilde{\Lambda})$ is a cluster point the existence of an infinite subsequence $\{t_k\}$ such that $(\theta(t_k), R(t_k), \Lambda(t_k)) \in B(\tilde{\theta}, \tilde{R}, \tilde{\Lambda}, \Delta) \subset \overline{D}_M$ immediately follows. The boundedness of $\xi$ is secured by M1, M2, and S2. A similar approach is used in [92].
Appendix B

Initialization Software Description

B.1 Description of Files

SetupInit.m

The setup file consists of a number of initial values and constants required to run Init.m. The number of states nx, the initial values of states and output, the scale factor for the sampling period, scalingTs, and the down-sampling factor for data saving all work as their counterparts in the RPEM setup. The main tuning parameters here are P_factor (the magnitude of the diagonal elements of the initial P matrix), R1_x, R1_theta, and R2 (the covariance matrices for the process and measurement disturbances).

Init.m

Contains the Kalman based initialization algorithm. The user is required to give polynomial orders of the states and inputs (polynomialOrders), the indices of the polynomial elements not to be used in the model structure (notUsedIndices) and an initial parameter vector (theta_0). The algorithmic loop runs the algorithm and saves signals and parameters for display of results. Finally the original parameters are calculated from the scaled ones.

reInitiate.m

The file is used for setup of a re-run of one of the algorithms. It sets the initial parameters to the final parameters of the previous run, and enables a change of model structure through a change of polynomialOrders or notUsedIndices. However, the change of model structure is usually not utilized when reInitiate.m is used to setup the parameters from the initialization algorithm for use in the RPEM algorithm.
B. Initialization software description

**h.m.m**

The model output equation is described in this file. As the output equation is restricted to the form

$$y(t) = x_1(t)$$

(B.1)

the file should normally not be changed if the initialization algorithm is to be used. Otherwise, the file works exactly as in the RPEM case. The RPEM allows for use of known nonlinear output function, cf. [95]

**dhdx.m.m**

Calculates the derivative of the output equation with respect to the state vector. For a linear output equation like (B.1) this implies that the output from the function will be a constant vector. As (B.1) is used when initializing the derivative will then always be \((1 \ 0 \ldots \ 0)\).

**F.m.m**

This function calculates the system matrix (6.8) as a function of the input \(u\) and the state estimation \(\xi\) obtained by differentiation of the measured output.

**dfdx.m.m**

Derivative of the RHS of the original system equation with respect to the states \(x\). This function is identical to the one used with the RPEM algorithm.

**differentiateY.m**

Estimates the state vector by repeated differentiations of the measured output as described in (6.4).

**GenerateIndices.m**

Generates the indices of the regressor vector. The user provides the maximum polynomial orders of each state and input (polynomialOrders), and the indices that correspond to the polynomial elements that are not to be used in the model structure (notUsedIndices). *GenerateIndices.m* then generates the indices of the regressor polynomial as all combinatorial combinations of the polynomial orders excluding the indices supplied by notUsedIndices.

**B.2 Examples**

To use the software implementation of the Kalman filter based initialization algorithm (6.10) the following steps are required.
B.2. Examples

B.2.1 Initialization Setup

The preparation for the identification run requires that the user

1. Modifies the output equation and the corresponding derivative of the underlying ODE model, as given by \( h_{m,m} \) and \( dhdx_{m,m} \). The function \( h_{m,m} \) implements the output equation of the model. Note that this function is allowed to be a nonlinear function of the state and input. The function is not allowed to be dependent on the estimated parameters, it must be known a priori. Note also that the derivative of the function, with respect to the estimated state, needs to be supplied in the function \( dhdx_{m,m} \).

2. Provides further input data in the script \( \text{SetupInit.m} \). The parameters that define the data generation are directly written into this script. These parameters define the dimension of the system, the initial value used in the ODE model, the size of the initial value of the \( P \)-recursion of (6.10), the magnitude of the covariance matrices for the state and parameter vector system disturbances, \( (R_{1,x} \) and \( R_{1,\theta} \) respectively), the magnitude of the measurement noise covariance, \( R_2 \), the scale factor \( \alpha \), as well as the down-sampling period used to avoid too large logs during long runs with high degree models.

3. Executes \( \text{SetupInit.m} \). This loads the necessary parameters into the MATLAB workspace.

Example 3. The system

\[
\begin{align*}
\dot{x}(t) + x(t) + (2 + u(t))x(t) &= u(t) \\
y(t) &= x(t) + e(t)
\end{align*}
\]

is to be identified with a second order model, which implies that the true parameter and corresponding regressor vectors are

\[
\theta_0 = (0 \quad 1 \quad -1 \quad 0 \quad -2 \quad -1 \quad 0 \quad 0)^T \quad (B.3)
\]

\[
\varphi(x,u) = (1 \quad u \quad x_2 \quad x_2u \quad x_1 \quad x_1u \quad x_1x_2 \quad x_1x_2u)^T \quad (B.4)
\]

The scale factor \( \alpha \) is selected equal to 2. The magnitude of the state and parameter disturbance covariance are selected to \( R_{1,x} = 0.01 \) and \( R_{1,\theta} = 0.001 \) respectively, while the measurement covariance matrix \( R_2 \) is selected equal to 1. The initial value of the \( P \)-recursion is selected equal to 0.1. The functions \( h_{m,m} \) and \( dhdx_{m,m} \) become

\[
\begin{align*}
h_{m,m} &\quad \text{function} \ [h_m]=h_m(x_m,u); \\
&\quad h_m=x_m(1,1);
\end{align*}
\]

and
B. Initialization software description

\[ \text{dhdx}_m \text{.m} \]  
\[
\text{function } [\text{dhdx}_m]=\text{dhdx}_m(x,u); \\
\quad \text{dhdx}_m=[1 \ 0];
\]

respectively. Further, the setup script \texttt{SetupInit.m} becomes

\[
\begin{align*}
\text{nx}=2; \\
\text{x}_m_0=[1 \ 0]'; \\
\%
\%
% \text{Remaining initial values}
%
\%
\text{y}_m_0=[\text{x}_m_0(1)]; \\
\text{scaleFactorP}=1e-1; \\
\text{R1}_x=1e-1; \quad \% \text{Scaling factor for state disturbances}
\text{R1}_\theta=1e-3; \quad \% \text{Scaling factor for parameter disturbances}
\text{R2}=1e-2; \quad \% \text{Scaling factor for measurement noise}
\%
\%
% \text{Parameters}
%
\%
\text{downSampling}=10; \quad \% \text{The down-sampling factor used when data}
\quad \% \text{from the run is saved}
\text{scalingTs}=2; \quad \% \text{The scale factor with which the sampling}
\quad \% \text{period is multiplied during identification}
\end{align*}
\]

Finally the user executes \texttt{SetupInit.m} in the MATLAB command window

\[
>> \text{setupInit}
\]

\[
>>
\]

B.2.2 Command Window Operation

In order to perform an identification run the user is required to execute and provide input to the script \texttt{Init.m}. The execution of this script makes use of four additional functions, implementing the polynomial model applied for modeling of the right hand-side (RHS) of the ODE. These functions are \texttt{F.m.m}, \texttt{dfdx.m.m}, \texttt{differentiateY.m} and \texttt{GenerateIndices.m}. The latter function generates the exponents of all factors of all terms of the polynomial expansion. The generation of these indices involves nested loops. They are therefore calculated in advance and used in repetitive calls in the form of a table.

To identify the system, the user is required to

1. Execute the script \texttt{Init.m}
2. Provide the degrees of the polynomial model (`polynomialOrders`) when prompted. The `polynomialOrders` variable is a column vector with the first element corresponding to the maximal degree of $x_1$, the second element corresponding to the maximal degree of $x_2$ and so on. The last element corresponds to the maximum degree of the derivative of highest degree of the last input signal component. In the present example, `polynomialOrders = [1 2 3]'` would mean that the highest degree term of the polynomial expansion is

$$\theta_{123}x_1x_2^2u^3. \quad (B.5)$$

3. Provide a list of indices that are not to be used (`notUsedIndices`) by the algorithm. The indices exclude terms in the polynomial expansions. Providing an empty matrix ([ ]) indicates that no terms shall be excluded. The list of not used indices are to be provided as rows in a matrix, where the number of rows equals the number of terms that are to be excluded from the model. In the present example `notUsedIndices = [0 0 0; 1 1 1]` would mean that the terms $\theta_{000}$ and $\theta_{111}x_1x_2u$ are to be excluded from the model.

4. Provide the initial parameter vector `theta0`. Note that unlike in the dynamic RPEM case, this parameter vector does not necessarily need to correspond to a linearized system with all poles within the unit circle. It is for example perfectly all right to choose `theta0=0`. However, when using the initialization algorithm in combination with the dynamic RPEM it is important to check that the final parameter vector obtained with Init.m corresponds to a linearized model with poles within the stability region as described in section 6. If this is not the case it will not be possible to use the obtained parameter vector for initializing the RPEM. Observe that if the scale factor of the sampling period is to be used for the RPEM, it also needs to be included during the initialization, so that parameters for the right optimization problem are generated.

**Example 4.** In this example the same system as in Example 3 is studied. The algorithm is in this example initialized with

$$\hat{\theta}(0) = (0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0)^T \quad (B.6)$$

This corresponds to the regressor vector

$$\varphi(x,u) = (1 \ u \ x_2 \ x_2u \ x_1 \ x_1u \ x_1x_2 \ x_1x_2u)^T \quad (B.7)$$

In this example comments and explanations have been added. To distinguish these from the actual commands the comments are in italics. The command sequence applied in the MATLAB command window is

```matlab
>> Init
ans =
Input polynomialOrders and notUsedIndices
```
- The script asks for the max degrees of states and inputs
K>> polynomialOrders=[1 1 1],
\[
\text{polynomialOrders} = \\
1 \\
1 \\
1
\]
K>> notUsedIndices= [ ]
\[
\text{notUsedIndices} = \\
[ ]
\]
K>> return
- The script returns the degrees of all included terms, 
input degrees to the right
\[
\text{allIndices} = \\
0 0 0 \\
0 0 1 \\
0 1 0 \\
0 1 1 \\
1 0 0 \\
1 0 1 \\
1 1 0 \\
1 1 1
\]
- The script asks for an initial parameter vector
K>> \text{ans} = 
\[
\text{Input theta_0} \\
\text{K>> theta_0=zeros(8,1)} \\
\text{theta_0} = \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\]
K>> return
- The script displays the fraction of the processing 
that is completed.
\[
\text{percentReady} = \\
100
\]
\[
\text{LinearizedPoleRadii} = \\
0.9956 \\
0.9956
\]
- The script displays the identified parameters, 
scaled parameters to the left
\[
\text{ans} = 
\]
B.2. Examples

0.0000  0.0000
0.2450  0.9798
-0.4900 -0.9800
-0.0004 -0.0007
-0.4900 -1.9600
-0.2449 -0.9797
-0.0002 -0.0003
-0.0006 -0.0012
>>

The estimated parameters of the left column are scaled parameters. The right column contain parameters that are recomputed to correspond to the original sampling period, cf. (3.21). Note that the exact result depends on the generated input signal. This may differ between systems and execution occasions since the seed for the random number generator may differ. Hence, slight variations of the estimated parameters are normal.

B.2.3 Re-initiation, Multiple Scans and Iterative Refinement

The script Init.m produces initial parameters for the RPEM. To run the RPEM from the end results of the initialization algorithm, the function reInitiate.m is required. In order to perform an RPEM run, the user is required to

1. Run the script reInitiate.m. That script prompts the user for the variables polynomialOrders and notUsedIndices. The parameter vector at the end of the run, together with the previous and new degrees, are then used to re-initiate all relevant quantities of the RPEM.

2. Run the SetupRPEM.m file, as described in Section 6 of [99].

3. Rerun RPEM.m. Note that the RPEM does not need to prompt the user for any further information this time.

Here, reInitiate.m is used in exactly the same way as described above. A more detailed description of the use of the reInitiate command can be found in Section 6.3 of [99]. Note, however, that the model structure is normally not changed when the initialization algorithm is used in combination with the RPEM, as the initial parameters are given for a specific model structure. By changing the model structure between the initialization and RPEM algorithms, the benefits of an initialization algorithm may be lost or significantly reduced.
B. Initialization software description
Proof of Theorem 2 (Chapter 7)

It is first noted that the conditions of Lemma 5 are fulfilled. This follows from $A_1$, the fact that (7.8) is quadratic and since the disturbances enter as known time variations in the criterion. The condition $A_2$ is needed to make the optimization problem well posed, with a set of minimum values. Next, the control is computed for the two cases.

C.1 The Optimal Control for $J_{\text{extended}}$

First consider the control problem corresponding to $J_{\text{extended}}$. Using

$$\bar{L}(\bar{x}(t), u(t), t) = \| y_{ref}(t) - (C \quad 0)\bar{x}(t) \|_{Q_1}^2 + \| (0 \quad B_1)\bar{x} \|_{Q_2}^2$$

the following quantities can be computed using (7.5) and (7.8)

$$\bar{L}_x^T(\bar{x}(t), u(t), t) = -2 \begin{pmatrix} C^T \\ 0 \end{pmatrix} Q_1 (y_{ref}(t) - \hat{y}(t))$$

$$+ 2 \begin{pmatrix} 0 \\ B_1^T \end{pmatrix} Q_2 (0 \quad B_1) \ddot{x} \equiv \begin{pmatrix} \bar{L}_x^T(\bar{x}(t), u(t), t) \\ \bar{L}_z^T(\bar{x}(t), u(t), t) \end{pmatrix}$$

$$\bar{L}_u(\bar{x}(t), u(t), t) = 0$$

$$\bar{f}_x^T(\bar{x}(t), u(t), t) = \begin{pmatrix} f_x^T(\bar{x}(t), u(t), t) \\ f_z^T(\bar{x}(t), u(t), t) \end{pmatrix}$$

$$\bar{f}_u(\bar{x}(t), u(t), t) = B$$

The shift matrix $D$ is given by

$$D = \begin{pmatrix} D_1 & 0 & \cdots & 0 \\ 0 & D_2 & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & D_k \end{pmatrix}$$
where each submatrix $D_i, i = 1, \ldots, k$, is a $(T_{u_i} \times T_{u_i})$ overdiagonal matrix

$$D_i = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \cdots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 0 \end{pmatrix}, i = 1, \ldots, k. \quad (C.7)$$

Also introduce the Lagrange multipliers

$$\bar{\lambda}(t) = \begin{pmatrix} \lambda^T_x(t) & \lambda_{1,1}(t) & \ldots & \lambda_{1,T_{u_1}}(t) & \ldots & \lambda_{k,1}(t) & \ldots & \lambda_{k,T_{u_k}}(t) \end{pmatrix}^T \quad (C.8)$$

corresponding to the extended state vector (7.3).

Note first that (7.17) in combination with (C.3) gives that

$$\lambda_{i,T_{u_i}}(t) = 0, \ i = 1, \ldots, k, \ \forall t. \quad (C.9)$$

Using equations (7.14) and (C.4) it follows that

$$\lambda_{i,j}(t) = \lambda_{i,j-1}(t+1), \ i = 1, \ldots, k, \ j \geq 2, \ \forall t \quad (C.10)$$

and consequently it follows from (C.9) that

$$\lambda_{i,j}(t) = 0, \ i = 1, \ldots, k, \ j \geq 2, \ \forall t. \quad (C.11)$$

The equations for $\lambda_{i,2}, \ i = 1, \ldots, k$ are central for the computation of the optimal control that follows.

For simplicity the notation is hereafter reduced, e.g. $\tilde{L}_x(x(t), u(t), t)$ is replaced by $L_x(t)$. Similar notations are used for the other components of the Euler-Lagrange equations.

To get an expression for $\bar{\lambda}(t)$, (7.14) is iterated, using (7.16) and (C.2)-(C.4), resulting in

$$\bar{\lambda}(N - p) = \begin{pmatrix} \lambda_x(N - p) \\ \lambda_{1,1}(N - p) \\ \lambda_{1,2}(N - p) \\ \vdots \\ \lambda_{k,1}(N - p) \\ \lambda_{k,2}(N - p) \\ \vdots \end{pmatrix} = \begin{pmatrix} \tilde{L}_x^T(N - p) \\ \tilde{L}_{z_{1,1}}(N - p) \\ \tilde{L}_{z_{2,1}}(N - p + 1) \\ \vdots \\ \tilde{L}_{z_{k,1}}(N - p) \\ \tilde{L}_{z_{k,1}}(N - p + 1) \\ \vdots \end{pmatrix}.$$
With the row vector with a 1 in the position corresponding to \(\bar{y}_{\text{ref}}(i)\), the LHS of (C.14) can then be calculated with a pre-multiplication of (C.2)

\[
\begin{align*}
&f_{z_{1,1}}^T(N-p)\left(\bar{L}_x^T(N-p+1) + \sum_{i=1}^{p-2} \left(\prod_{j=i}^{p-2} f_x^T(N-j-1)\right) \bar{L}_x^T(N-i)\right) \\
&f_{z_{1,1}}^T(N-p+1) \left(\bar{L}_x^T(N-p+2) + \sum_{i=1}^{p-3} \left(\prod_{j=i}^{p-3} f_x^T(N-j-1)\right) \bar{L}_x^T(N-i)\right) \\
&\vdots \\
&f_{z_{k,1}}^T(N-p)\left(\bar{L}_x^T(N-p+1) + \sum_{i=1}^{p-2} \left(\prod_{j=i}^{p-2} f_x^T(N-j-1)\right) \bar{L}_x^T(N-i)\right) \\
&f_{z_{k,1}}^T(N-p+1) \left(\bar{L}_x^T(N-p+2) + \sum_{i=1}^{p-3} \left(\prod_{j=i}^{p-3} f_x^T(N-j-1)\right) \bar{L}_x^T(N-i)\right) \\
&\vdots
\end{align*}
\]

(C.12)

From the result (C.12) the expression for \(\lambda_{a,2}(t)\) is obtained as

\[
\lambda_{a,2}(t) = \bar{L}_{z_{a,1}}(t+1) + f_{z_{a,1}}^T(t+1) \left(\sum_{i=t+2}^{N-1} \left(\prod_{\ell=t+2}^{i-1} f_x^T(\ell)\right) \bar{L}_x^T(i)\right),
\]

\[\forall t < N, \ a = 1, \ldots, k. \tag{C.13}\]

The equation (C.13) holds for \(t = N - 1\) since \(\bar{L}_{z_{a,1}}(N) = 0\) must hold from consistency considerations. This follows since \(0 = \lambda_{a,2}(N - 1) = \lambda_{a,1}(N)\).

The equations (C.11) are then utilized, to set \(\lambda_{a,2}(t)\) of (C.13) equal to zero. This results in the equation

\[
\bar{L}_{z_{a,1}}(t+1) = -f_{z_{a,1}}^T(t+1) \left(\sum_{i=t+2}^{N-1} \left(\prod_{\ell=t+2}^{i-1} f_x^T(\ell)\right) \bar{L}_x^T(i)\right),
\]

\[\forall t < N, \ a = 1, \ldots, k. \tag{C.14}\]

The LHS of (C.14) can then be calculated with a pre-multiplication of (C.2) with the row vector with a 1 in the position corresponding to \(z_{a,1}(t+1)\). The result is

\[
\bar{L}_{z_{a,1}}(t+1) = 2(Q_2)_{a} \ \text{vec}(z_{i,1}(t+1)), \ \forall t < N, \ a = 1, \ldots, k, \tag{C.15}\]

where \((Q_2)_{a}\) denotes the \(a\):th row of \(Q_2\). It also immediately follows from (C.2) that

\[
\bar{L}_x^T(i) = -2C^TQ_1(y_{\text{ref}}(i) - \bar{y}(i)). \tag{C.16}\]

The equations (C.14)–(C.16) can be combined for all \(a\) into the systems of
C. Proof of Theorem 2

Equations

\[ Q_2 \text{vec}(z_{a,1}(t+1)) = \]
\[ \sum_{a=1,\ldots,k}^{N-1} \left( \prod_{t=t+2} \left( f_x^T(\ell) \right) C^T Q_1(y_{ref}(i) - \hat{y}(i)) \right), \quad \forall t < N. \]

Since \( Q_2 \) is nonsingular (by \( A_3 \)), the control follows from (C.17), using the definition (7.4)

\[ \text{vec}(u_a(t - T_{u_a})) = \]
\[ \sum_{a=1,\ldots,k}^{N-1} \left( \prod_{t=t+1} \left( f_x^T(\ell) \right) C^T Q_1(y_{ref}(i) - \hat{y}(i)) \right), \]
\[ \forall t < N - 1. \]

In the SISO case, the solution collapses to the one of [98], as expected.

C.2 The Optimal Control for \( J_{direct} \)

Define

\[ \ddot{u}(t) = (u_1(t - T_{u_1}) \ldots u_k(t - T_{u_k}))^T. \]

With this use of input vector (7.10) represents the standard problem of Lemma 5. The equations (7.14)-(7.17) give that

\[ L_x^T(x(t), \ddot{u}(t), t) = -2C^T Q_1(y_{ref}(t) - \hat{y}(t)) \]
\[ L_u^T(x(t), \ddot{u}(t), t) = 2Q_2 \ddot{u}(t) \]

Solving (C.21) for \( \ddot{u}(t) \), using the fact that \( Q_2 \) is positive definite, results in

\[ \ddot{u}(t) = \frac{1}{2} Q_2^{-1} L_{\ddot{u}}(t) = \frac{1}{2} Q_2^{-1} \lambda^T(t + 1)f_{\ddot{u}}(t). \]

The equation (7.14) can then be iterated starting with (7.16), resulting in

\[ \lambda(t) = \sum_{i=t}^{N-1} \left( \prod_{j=t}^{i-1} f_x^T(j) \right) L_x^T(i) \]

Combining (C.20), (C.22) and (C.23) then gives the control

\[ \ddot{u}(t) = Q_2^{-1} f_{\ddot{u}}^T(t) \left( \sum_{i=t+1}^{N-1} \left( \prod_{j=t+1}^{i-1} f_x^T(j) \right) C^T Q_1(y_{ref}(i) - \hat{y}(i)) \right), \quad t < N - 1 \]

(C.24)
C.3 Equivalence of $J_{direct}$ and $J_{extended}$

The Lagrange multipliers have now been eliminated from the Euler-Lagrange equations. It can easily be seen that (C.18) and (C.24) are identical, since

$$\bar{u}(t) = \text{vec}(u_a(t - T_{ua}))$$

and since

$$f_\bar{u}(t) = \text{vec}(f_{z_{a,1}}(t))$$

When inserted in the criterion functions (7.10) and (7.8), this then proves Theorem 1.

Remark: Note that the controls depend on future values of the states, not yet known when the control is to be computed. This is normal for systems with delays, and is solved by the use of predicted states calculated from the information known at the time of control computation. Note also that although the controls have been expressed in terms of the states, an implicit dependence of the inputs still exist in the partial derivatives of $f$. Hence an iterative search of the optimal control signal will be necessary.
C. Proof of Theorem 2
Appendix D

MPC software description

D.1 Description of Files

SetupSim.m

The setup file consists of a number of constants, buffers, and initial values for variables used in the files Simulator.m and controllerShell.m. The most important user chosen parameters are:

- **Model parameters**: This is where the model parameters and signal scale factors are set.
- **Problem size**: Number of inputs and data points.
- **Sampling periods**: For simulation and control signal calculation. For details, see the description of Simulator.m below
- **Sampling and time constants**: The sampling constants can be adjusted to change the number of samples used at a certain input. By default nominalSamplingPeriod=5, and nominalFlow=3000. This means that if the flow is 3000 the sampling period inside the controller will be 5. The sampling period for other flows are normalized using these values. The time constants include the delays (in samples) of each input and disturbance, and the length of the control horizon. These values need to be matched so that at a certain input (e.g. the nominal input) the delay for each signal is equal to the corresponding delay times the input dependent sampling period (for that input).
- **Constants for pump flow**: These parameters are for converting the flow to the pump speed as fed to the system. Further, the lower limit for this speed is set (given in percent of full speed).
- **Variables for initialization of the system**: These are the initial values for the signals in the simulator. By default they are given on the form
measured signal multiplied by the scale factor. In this way the same initialization can be used regardless of model and the corresponding scaling. For example, since \( y_0 \) and \( v_2^0 \) are temperatures, \( y_0 \) and \( v_2^0 \) are the initial temperatures (in °C) multiplied by their respective scale factors.

- **Variables for initialization of the controller:** Similar to the above described initialization, but here it is the values used inside `controllerShell.m`. The values will be gradually replaced by sampled data from the simulator as the simulation starts.

- **Reference signal:** The reference signal is a vector containing the desired output (outlet temperature) corresponding to each data point.

- **Disturbances:** Can be loaded from a .mat file or set constant, depending on what the user wants to study.

- **Controller constants:** These are tuning variables of the optimization algorithm (7.19) [24] to be chosen by the user. The parameter \( k \) affects the step size in the optimization algorithm, \( \text{tol} \) decides how small the Hamiltonian needs to be before the optimization is considered complete, \( \text{mxit} \) is the maximum number of iterations the algorithm can do without being interrupted (as a precaution, in case the algorithm does not converge) and \( \rho \) is the control signal penalty of the criterion function.

### Simulator.m

This file simulates the system, presently corresponding to the solar collector and heat storage tanks as described in the introduction. To deal with the input dependent time delays the values of \( u_0, v_1, \) and \( v_2 \) used for simulation are taken from the vectors \( u\_save, v1\_save, \) and \( v2\_save \) respectively, depending on the input signal (flow).

In the present version of the software the system dynamics need to be set in the beginning of the simulator. By default the system and the continuous time model used in the controller are the same. The simulator uses a small step size \( Ts\text{Small} \) for updating of the discretized system equations. The sampling period \( Ts \), which determines the instances for calculation of the next control sequence (by calling `controllerShell.m`), is generally larger (typically \( Ts\text{Small}=1 \) and \( Ts=5 \)). After calculating the new control signal, the most recent values of the input \( u \), the disturbances \( v_1 \) and \( v_2 \), the output \( y \), and the sampling time are saved. In addition the computed state inside the controller (corresponding to \( x_0 \)) is saved in \( y\text{Hat}_\text{save} \) (note that the output equation gives that \( y = x_1 \)).

### controllerShell.m

This function is called by the simulator (or possibly the control computer), and handles input signal dependent sampling of the data (calling the `flowSample.m` function), calculating the optimal control sequence (by calling `optimalController.m`), resampling to the regular sampling period, converts the control signal (flow to pump speed that is the true control signal), and make
sure the control signal (pump speed) is within the allowed limits. Finally the initial state estimate $x_0$ is updated using \texttt{updateS.m}.

\textbf{optimalController.m}

This function contains the implementation of the gradient search optimization algorithm as described in [24]. From this file other functions (\texttt{dLdu.m}, \texttt{dLdx.m}, \texttt{f.m}, \texttt{df.m}, and \texttt{calculatePhi.m}) that describe model and criterion functions are called. To enable penalization of the derivative of the input, the state space description of the model has been extended to containing the system input. The nomenclature parallels that of chapter 2 of [24], which should make it relatively straight forward to understand what each command in the algorithm does.

There is a possibility of adding a hard limit (\texttt{limit}) to restrict the size of the derivative of $u$, if necessary. This option was utilized mainly during the development phase of the software generation. By default, however, the limit is set to $10^6$, which means it is usually not active.

\textbf{flowSample.m}

This script resamples the data proportionally to the inverse of the input to obtain a time delay that is constant in the number of samples. The buffers $uS$, $v1S$, and $v2S$ contain the input signal dependent sampled input and disturbance signals respectively. Further $tS$ is the next sampling instant for which the control signal is to be calculated, $dS$ is the sampling period, and $nextS$ contains all instants for which the control signal should be calculated. The script is called from \texttt{controllerShell.m}

\textbf{updateS.m}

This function updates the model state equation by integration, given the previous values of the state, the input, the disturbances and the step size. The equation is typically on the form (7.23)-(7.24), but could be altered by the user. Note that this function is not based on the extended state space representation of the model, but of the original one. The function is called from \texttt{controllerShell.m}.

\textbf{dLdu.m}

This function calculates the derivative of the criterion function with respect to the input. As the optimal controller uses the extended state representation (7.26)-(7.27) it is with respect to the derivative of the input that $L(x, \dot{u}, \rho)$ is differentiated in this function. This function can easily be altered by the user if a different criterion function is desired. The function is called from \texttt{optimalController.m}.
dLdx.m

This function implements the derivative of the criterion function with respect to the state vector. The optimal controller in this software utilizes the extended state representation (7.26)-(7.27), so the derivative of $L(x, \dot{u}, y_{ref}, \rho)$ is with respect to both the system state and input. The function is called from `optimalController.m`.

f.m

This function updates the extended state equation by integration (7.26). Again, the state vector has been extended to containing the input, whereas the derivative of the input is used as a new input signal. This enables penalization of the derivative of the input. The structure of the equation is similar to that of `updateS.m`, but altered for the extended state representation. The function is called from `optimalController.m`.

df.m

This function implements the derivative of $f$ (cf. f.m above) with respect to extended states and $\dot{u}$. The function is called from `optimalController.m`.

calculatePhi.m

In this function $\phi(x_N)$ is the terminal constraint of the optimization algorithm. In the software package it is set to a similar quadratic function as for the rest of the criterion. This m-file calculates both $\phi$ and its derivative with respect to the extended state vector. It is here utilized that according to the output equation $y(t) = x_1(t)$, so that the reference signal $y_{Ref}$ can be compared to $x_1$ instead of $y$. The function is called from `optimalController.m`.

D.2 Examples

To run a simulation of the plant the following commands need to be executed in the MATLAB command prompt

```matlab
>> SetupSim
>> Simulator
```

As described earlier the setup file generates parameters required by the other functions, whereas the simulator calls the controller shell to obtain a new control signal. To study the results the regular plot commands in MATLAB are used. For example

```matlab
>> figure
>> plot(t_save,y_save/scalingY)
>> hold on
```
>> plot(t_save,yRef,'--')

generates a plot of the simulated outlet temperature from the system and the corresponding reference temperature. Note that \( y_{\text{save}} \) is only available at each sampling instant, while \( y_{\text{Ref}} \) is the same size as the data set. Further, the saving of the variables is done in their scaled version, and therefore need to be rescaled at plotting. Similarly

>> figure 
>> plot(t_save,u_save/scalingU)

>> figure 
>> plot(t_save,v1_save/scalingV(1))

>> figure 
>> plot(t_save,v2_save/scalingV(2))

generates plots of the flow, the solar radiation and the inlet temperature respectively. The results of a run with the default values of \texttt{SetupSim.m} can be seen in Fig. D.1-D.4.
Figure D.1: Outlet temperature and reference.

Figure D.2: Flow [L/h]
D.2. Examples

Figure D.3: Solar radiation [$W/m^2$]

Figure D.4: Inlet temperature to the solar collectors [$^\circ C$].
Bibliography


[56] ———, 2003, personal communication.


