Helsinki University of Technology Control Engineering

Espoo 2009 Report 162

SIMULATION ASSISTED PERFORMANCE OPTIMIZATION OF LARGE-SCALE MULTIPARAMETER TECHNICAL SYSTEMS

Kalle Halmevaara

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D´HELSINKI

Helsinki University of Technology Control Engineering

Espoo September 2009 Report 162

SIMULATION ASSISTED PERFORMANCE OPTIMIZATION OF LARGE-SCALE MULTIPARAMETER TECHNICAL SYSTEMS

Kalle Halmevaara

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Faculty of Electronics, Communications and Automation, for public examination and debate in Auditorium AS2 at Helsinki University of Technology (Espoo, Finland) on the 23th of October, 2009, at 12 noon.

Helsinki University of Technology

Faculty of Electronics, Communications and Automation

Department of Automation and Systems Technology

Distribution:
Helsinki University of Technology
Department of Automation and Systems Technology
P.O. Box 5500
FI-02015 TKK, Finland
Tel. +358-9-451 5201
Fax. +358-9-451 5208
E-mail: control.engineering@tkk.fi
http://autsys.tkk.fi/

ISBN 978-952-248-098-9 (printed)
ISBN 978-952-248-099-6 (pdf)
ISSN 0356-0872

Yliopistopaino
Helsinki 2009

Available on net at http://lib.tkk.fi/Diss/2009/isbn9789522480996

ABSTRACT OF DOCTORAL DISSERTATION	HELSINKI UNIVERSITY OF TECHNOLOGY P.O. BOX 1000, FI-02015 TKK http://www.tkk.fi

Author Kalle Halmevaara

Name of the dissertation

Simulation assisted performance optimization of large-scale multiparameter technical systems

Manuscript submitted	31.3.2009	Manuscript revised	3.9.2009
Date of the defence	23.10.2009		
☒ Monograph		☐ Article dissertation (summary + original articles)	

Faculty	Faculty of Electronics, Communication and Automation
Department	Department of Automation and Systems Technology
Field of research	Control Engineering
Opponent(s)	Prof. Kauko Leiviskä and Prof. Hannu Koivisto
Supervisor	Prof. Heikki Koivo
Instructor	Prof. Heikki Hyötyniemi

Abstract

During the past two decades the role of dynamic process simulation within the research and development work of process and control solutions has grown tremendously. As the simulation assisted working practices have become more and more popular, also the accuracy requirements concerning the simulation results have tightened. The accuracy improvement of complex, plant-wide models via parameter tuning necessitates implementing practical, scalable methods and tools operating on the correct level of abstraction.

In modern integrated process plants, it is not only the performance of individual controllers but also their interactions that determine the overall performance of the large-scale control systems. However, in practice it has become customary to split large-scale problems into smaller pieces and to use traditional analytical control engineering approaches, which inevitably end in suboptimal solutions.

The performance optimization problems related to large control systems and to plant-wide process models are essentially connected in the context of new simulation assisted process and control design practices. The accuracy of the model that is obtained with data-based parameter tuning determines the quality of the simulation assisted controller tuning results. In this doctoral thesis both problems are formulated in the same framework depicted in the title of the thesis. To solve the optimization problem, a novel method called Iterative Regression Tuning (IRT) applying numerical optimization and multivariate regression is presented. IRT method has been designed especially for large-scale systems and it allows the incorporation of domain area expertise into the optimization goals.

The thesis introduces different variations on the IRT method, technical details related to their application and various use cases of the algorithm. The simulation assisted use case is presented through a number of application examples of control performance and model accuracy optimization.

Keywords	Controller tuning, process simulation, large-scale technical systems, numerical optimization		
ISBN (printed)	978-952-248-098-9	ISSN (printed)	0356-0872
ISBN (pdf)	978-952-248-099-6	ISSN (pdf)	
Language	English	Number of pages	120
Publisher	Helsinki University of Technology, Department of Automation and Systems Technology		
Print distribution	Helsinki University of Technology, Department of Automation and Systems Technology		
☒ The dissertation can be read at http://lib.tkk.fi/Diss/2009/isbn9789522480996			

VÄITÖSKIRJAN TIIVISTELMÄ	TEKNILLINEN KORKEAKOULU PL 1000, 02015 TKK http://www.tkk.fi

Tekijä Kalle Halmevaara

Väitöskirjan nimi

Simulointiavusteinen laajojen moniparametristen teknisten systeemien suorituskyvyn optimointi

Käsikirjoituksen päivämäärä 31.3.2009	Korjatun käsikirjoituksen päivämäärä 3.9.2009
Väitöstilaisuuden ajankohta 23.10.2009	
☒ Monografia	☐ Yhdistelmäväitöskirja (yhteenveto + erillisartikkelit)

Tiedekunta	Elektroniikan, tietoliikenteen ja automaation tiedekunta
Laitos	Automaatio- ja systeemitekniikan laitos
Tutkimusala	Systeemitekniikka
Vastaväittäjä(t)	Prof. Kauko Leiviskä ja prof. Hannu Koivisto
Työn valvoja	Prof. Heikki Koivo
Työn ohjaaja	Prof. Heikki Hyötyniemi

Tiivistelmä

Kahden viimeisen vuosikymmenen aikana dynaamisen prosessisimuloinnin merkitys on kasvanut huimasti erilaisissa prosessien ja säätöjärjestelmien tutkimus- ja suunnittelutehtävissä. Simulointiavusteisten työskentelytapojen yleistymisen myötä simulointitulosten tarkkuusvaatimukset ovat kasvaneet. Kompleksisten, tehdasmittakaavan dynaamisten mallien tarkkuuden parantaminen malliparametreja virittämällä edellyttää mallinnusympäristöiltä helppokäyttöisiä, laajoihin tarkasteluihin skaalautuvia oikean abstraktiotason menetelmiä ja työkaluja.

Nykyaikaisissa integroiduissa prosessilaitoksissa yksittäisten säätimien toiminnan lisäksi säätöpiirien välisillä ristikkäisvaikutuksilla on merkittävä vaikutus laajojen säätöratkaisujen kokonaissuorituskykyyn. Tästä huolimatta käytännössä usein tyydytään tarkastelemaan säätöjärjestelmiä laajojen kokonaisuuksien sijasta useassa pienemmässä osassa analyyttisin perinteisin säätöteknisin menetelmin, mikä johtaa väistämättä suboptimaalisiin ratkaisuihin.

Laajojen säätöjärjestelmien suorituskyvyn ja prosessimallien tarkkuuden optimointi nivoutuvat oleellisesti yhteen uusissa prosessi- ja säätösuunnittelun simulointiavusteisissa työtavoissa. Mitä tarkemmaksi malli voidaan virittää havaintoihin perustuen, sen parempiin tuloksiin simulointiavusteisella säätöjen virityksellä päästään. Tässä väitöskirjassa molemmat ongelmat formuloidaan samaan, väitöskirjan nimen kuvaamaan kehykseen. Kyseisen optimointiongelman ratkaisemiseksi esitellään Iterative Regression Tuning (IRT) menetelmä, joka on numeerista optimointia ja monimuuttujaregressiota hyödyntävä algoritmi. IRT menetelmä on suunniteltu erityisesti laajojen kokonaisuuksien tarkasteluun ja se mahdollistaa sovelluskohteeseen liittyvän asiantuntemuksen huomioonottamisen viritystavoitteita määriteltäessä.

Väitöskirjassa esitellään algoritmista eri variaatioita ja niiden soveltamiseen liittyviä yksityiskohtia sekä useita IRT menetelmän eri käyttötapauksia. Simulointiavusteista käyttötapaa malli- ja säätöparametrien virittämiseen demonstroidaan usean esimerkin avulla.

Asiasanat	Säätöjen viritys, prosessisimulointi, laajat tekniset systeemit, numeerinen optimointi		
ISBN (painettu)	978-952-248-098-9	ISSN (painettu)	0356-0872
ISBN (pdf)	978-952-248-099-6	ISSN (pdf)	
Kieli	Englanti	Sivumäärä	120

Julkaisija Teknillinen korkeakoulu, Automaatio- ja systeemitekniikan laitos

Painetun väitöskirjan jakelu Teknillinen korkeakoulu, Automaatio- ja systeemitekniikan laitos

☒ Luettavissa verkossa osoitteessa http://lib.tkk.fi/Diss/2009/isbn9789522480996

PREFACE

The last six years that I have worked in the Control Engineering Group at TKK and in the System Dynamics group at VTT have taught me a lot – scientifically, professionally and socially. It has also been a most enjoyable time and probably I will miss it later on.

First of all, I want to thank my supervisor Prof. Heikki Hyötyniemi for the creative ideas behind the thesis, for showing what genuine enthusiasm for making science is all about, and for his support and motivation during the past years. It has been a great pleasure to work with you both on research as well as educational activities.

I am also grateful to Prof. Heikki Koivo for being such an excellent head for the research group. The relaxed and secure working environment is largely due to your management skills. Also my colleagues both at TKK and at VTT deserve to be acknowledged – working together makes sense also in the scientific realm. Especially cooperation with Tamara Beltrame, Olli Haavisto, Kalle Kantola, Tommi Karhela, Pasi Laakso, Matti Lehtomäki, Matti Paljakka, Jouni Savolainen and Jean-Peter Ylén has been highly valuable. I also want to thank my friends and family for their support and encouragement.

The research projects related to the work were funded by TEKES, VTT, Fortum Nuclear Services, Metso Automation and Jaakko Pöyry. The Control Engineering Group supported financially the completion of this thesis. The further financial support received from the Walter Ahlström Foundation, the Jenny and Antti Wihuri Foundation, the 30th Anniversary Foundation of Neles Inc. and Automation Foundation is also highly appreciated.

Kalle Halmevaara

LIST OF SYMBOLS

A	Arbitrary symmetric positive definite matrix
D	Distance measure
F	Linear mapping matrix
F_W	Linear mapping matrix computed from whitened data
G	Process transfer function
G_{C}	Controller transfer function
G_{CL}	Closed loop transfer function
G_{F}	Filter transfer function
H	Hessian matrix
I	Identity matrix
J	Cost function
K	(Global) iteration index
K_P	Gain of the PID controller
L	Lipshitz constant
M	Arbitrary matrix
N	Latent variable dimension
Q	Quality measure matrix
Q_S	Scaled quality measure matrix
Q_W	Whitened quality measure matrix
R	Covariance matrix
S	Scaling matrix
T	Length of time series vector
T_I	Integration time of PID controller
W	Whitening matrix
a	Random variable
c	Loading vector related to output variables
d	Search direction
e	Error (signal) between target and observation
f_i	i^{th} column of matrix F
g	Scalar valued function
k	Number of samples, or discrete time index
m	Number of output variables (quality measures)
n	Number of input variables (decision variables)
p	Loading vector related to input variables

q	Quality measure vector
q^{-1}	Backward transfer operator in continuous time
s	Commensurable target unit vector, or Laplace variable
t	Continuous time variable
u	Control signal vector
v	Mixed variable vector
w	Weight vector
x	Input variable vector
y	Output or controlled variable vector
z	Latent variable, score value or principal component vector
E	Residual matrix
Φ	Complete set of eigenvectors or singular vectors
Γ	Vector of arbitrary scalar coefficients
Θ	Decision variable matrix
Θ_S	Scaled decision variable matrix
Θ_W	Whitened decision variable matrix
P	Diagonal matrix of ρ_i
Σ	Diagonal singular value matrix
Ψ	Output data oriented eigenvector set
α, β, χ	Arbitrary scalar coefficients
δ	Random perturbation vector
ε	Residual vector
ϕ	Basis vectors of N dimensional latent variable space
ϕ_i	i^{th} eigenvector, left singular vector, or column of ϕ
φ_i	i^{th} right singular vector
γ	Scalar coefficient for adjusting the length of the update step
λ	Eigenvalue
μ	Forgetting factor
θ	Decision variable (parameter) vector
ρ	Maximum deviation of θ from $\bar{\theta}$ in the local sampling
σ	Singular value or standard deviation
ω	Angle between vector
$\Re$	Feasible region
$\Re_S$	(Local) sampling domain

NOTATIONS

$\hat{x}$	Estimate of x
$\tilde{x}$	Best known value of x
$\overline{x}$	Nominal value or mean value of x
∇x	Gradient of x (column vector)
Δx	Deviation of x from $\overline{x}$
x'	Modified value of x
x^*	Optimal value of x
x^T	Transpose of x
$g(x)$	g is a function of x
$\mathrm{E}\{x\}$	Expectation value of x
$\mathrm{var}\{x\}$	Variance of x
$\mathrm{cov}\{x\}$	Covariance of x
$\lvert x \rvert$	Absolute value of x
$\lVert x \rVert$	Euclidean norm of x
$\{x : g(x) = \alpha\}$	Set containing the values of x which satisfy the condition $g(x) = \alpha$
$x \sim N_m$	x is random variable from m dimensional Gaussian distribution
x_i	i^{th} element of (column) vector x, or i^{th} column vector of matrix x
$\frac{dx}{dt}$	Differential of x with respect to t
$\frac{\partial x}{\partial t}$	Partial differential of x with respect to t

ABBREVIATIONS

AR	Autoregressive
CDF	Cumulative Distribution Function
CL	Closed Loop
CPA	Control Performance Assessment
CTMP	Chemi-Thermo-Mechanical Pulping
CV	Cross Validation
ECDF	Empirical Cumulative Distribution Function
ES	Extremum Seeking
GMV	Generalized Minimum Variance
GO	Global Optimization
IAE	Integral of Absolute value of Error
IL/ILC	Iterative Learning (Control)
IMC	Internal Model Control
IFT	Iterative Feedback Tuning
IRT	Iterative Regression Tuning
ISE	Integral of Squared Error
ITSE	Integral of Time-weighted Squared Error
LO	Local Optimization
LQG	Linear Quadratic Gaussian
LS	Least Squares
MIMO	Multiple Inputs, multiple outputs
MLR	Multiple Linear Regression
MPC	Model Predictive Controller
MRAC	Model Reference Adaptive Control
MSE	Mean Squared Error
MVR	Multivariate Regression
NIPALS	Nonlinear Iterative Partial Least Squares
OPC	Object Linking and Embedding technology for Process Control
PAS	Pure Adaptive Search
PDE	Partial Differential Equation
PI/PID	Proportional-Integral(-Derivative) controller
PLS	Partial Least Squares
PMO	Plant Model Oriented
PRS	Pure Random Search
RC	Repetitive Control
RIRT	Recursive Iterative Regression Tuning
RMSE	Root Mean Squared Error

SA	Simulated Annealing
SISO	Single input, single output
SQP	Sequential Quadratic Programming
STR	Self-Tuning Regulator
SVD	Singular Value Decomposition
UI	User Interface
VRFT	Virtual Reference Feedback Tuning
XML	Extensible Markup Language

CONTENTS

1 INTRODUCTION

Mathematical process modeling has always played an essential role in different phases of process and control engineering including research and development, design as well as operation of the plant and its control systems. The development of control theory has increased the call for dynamical process modeling and, vice versa, the results of rigorous dynamical analyses have contributed to the development of the control theory.

Before the 1940s most industrial process plants were operated almost completely manually without any *automatic process control* devices [56]. It was only in the 1940s and early 1950s that the first feedback controllers were introduced to the process industry due to an increasing economical pressure. At this stage controller design and controller tuning methodologies were still in their infancy and rough rules of thumb, based on the experience of plant personnel, were used for setting up automatic control strategies.

The development of industrial automation and control systems has been driven by the pursuit of economic efficiency, better end product quality, and growing environmental awareness in society. In the 1960s more sophisticated dynamical process analysis, advanced control theory and its applications were adopted to chemical engineering. The oil crisis in the 1970s forced the chemical engineers to increase the *process integration* of the plant design in order to save energy, which, for its part, made the process modeling and controller design even more complicated due to interacting process variables. At the same time interest in *systems engineering* that was trying to answer the question how to deal with or examine the complex industrial systems as single intertwined and interacting units started to gain ground. Later on, for example, *artificial intelligence* and *expert systems* became the successors to advanced control theory in the pursuit of mastering the complexity in highly developed industrial systems.

The introduction of modern control theory into the process industry required more careful dynamical analysis of the processes. At first the modeling concentrated on relatively simple systems that could be handled analytically since the computers existed only on the conceptual level in the early 1940s. However, elementary process models based on, for example, differential equations, were usually accurate enough for designing simple but still practical SISO controllers. Use of larger and more accurate dynamic plant-wide process models was adopted only when the development of advanced multivariable control structures began. The time needed for implementation and commissioning of such controllers could be reduced notably with simulator testing. Simulation also provided the possibility to detect the possible faults in the controller without a threat of damaging the process or causing a hazardous situation for the plant personnel. Nowadays dynamic simulation is widely used in controller design.

The rigorous *computer aided modeling* and *simulation* of *large-scale complex systems* could not kick off until computers had reached a sufficient level of sophistication in the 1980s. Since then *computational science* or *computationalism* has become the third corner stone of modern engineering along with scientific theory and experimental results. In practice this means that computer simulations are used for finding *numerical solutions* to problems instead of trying to find rigorous analytical solutions. In addition to various engineering applications modeling and simulation have been applied extensively also to many different areas, for example, natural sciences, economical research, etc. Nowadays weather forecasts, the design of mechanical constructions, business strategies, studies of biological systems, among others, all rest greatly on mathematical models and computed forecasts. Typically, models fit rather badly for other purposes than for the original target of application since the accuracy requirements in different applications can vary considerably. Consequently, modeling and simulation methods and software have been developed quite independently for different application domains.

Professional dynamic modeling and simulation software designated for the process industry started to be readily available only in the 1980s [41]. Nowadays there exists a large variety of simulation tools for a wide variety of purposes, from very general products to extremely specific professional tools. Within the process industry, simulation is often used as a substitute for or as a supplement to the experiments conducted on the real system. In some cases it is simply too expensive, time-consuming, laborious or dangerous to run experiments on the existing system. Typical applications are, for example, process and control system analysis and design, training and operator assistance, safety analysis, and production management. At present the number of simulation software suppliers is huge, for instance, there are more than a dozen software listed in [41] solely for the professional simulation of combustion or nuclear power plants.

The recent progression of software development, computer and computational science has naturally had an effect on the methods and tools of everyday engineering work. Also new challenges are emerging. For example, constructing impressive plant-wide process models is relatively straightforward for an experienced process engineer since typically the professional modeling software provide the user with extensive model libraries of different process components. The modeling environments, however, do not always support the data based model parameter identification that is a crucial operation if the accuracy of simulation results is important. The improvements of information technology leave questions open also what it comes to the design of plant-wide control systems. Even though it is beneficial to demonstrate the performance of a large complex control concept with a simulator before its implementation, the result of controller tuning will not be improved if the elementary design techniques from the 1940s are still applied. The increased computational power should be harnessed for addressing the controller design and tuning problems as well. Both model parameter optimization and control parameter tuning are example problems of managing modern large-scale complex technical systems for which new solutions can be found by adopting the computational approach.

1.1 Managing complexity in large scale industrial systems

Large biological, economical and social systems, such as the metabolism of bio organisms, the global stock market and cultural anthropology, are all examples of *complex systems*. The concept of complex system was initially associated with systems

whose performance in future is impossible to predict, in other words, to mathematically perfectly determined nonlinear systems that tend to express unpredictable and complex behavior starting from a well defined initial condition. These systems are nowadays more commonly called *chaotic systems* and the concept of *emergence* has become the distinguishing quality of complex systems. *Self-organization* is a form of emergence in which apparently intelligent global behavior emerges from a population of individual agents interacting only locally. For example, the collaboration of individuals in an ant colony looks organized, even though centralized control is not involved. Analyzing and modeling these emergent level phenomena is usually in the scope of complexity research. The *dimensional complexity* (e.g. the number of interacting individual agents) makes the analysis of these systems challenging. The scalability of analyzing methods is essential since, in general, the emergence and self-organization in complex systems takes place only when large enough entities are examined.

Traditionally, engineering has relied on *reductionism* whenever a big and challenging problem has been faced. Splitting a large problem into a number of easier tasks usually facilitates finding the solution. In this case the emergent phenomena are, however, completely neglected and one inevitably results in a non-optimal solution. Another stumbling block of engineers has been the attempt to model and analyze large systems in a rigorous bottom-up manner, i.e., incorporating each technical detail into the model and resulting in a complex replica of the original system that prevents any analytical examination. As proposed by Hyötyniemi [35] it is perhaps best to switch from a deterministic to a stochastic viewpoint when the size of the system grows enough. The essence (and the emergence) in large complex systems is best captured with a plant-wide *statistical approach* and it seems to be beneficial to examine the *degrees of freedom* in the system instead of the cohesive *constraints*.

Modern plant-wide control systems and detailed dynamic simulation models of industrial processes are good examples of complexity in technical systems. Although direct comparison of these systems to agent populations may feel artificial there exists certain resemblance. The inherent feedback couplings of the process models and the feedback loops of the control structures increase the interconnectedness of the systems and keep them in *dynamical balance*, but at the same time they make it hard to predict the performance of the system accurately. Furthermore, the *quality of performance* in both of the systems can be seen as an emergent phenomenon which cannot be analyzed or predicted in a piecewise manner.

In reality the industry and engineering offices often follow years behind the cutting edge of the research. Although hundreds of controller tuning methods have been published during the past decades solely for PI and PID controllers [62], the status at the factory floor level still remains relatively bad – many implemented controllers have been switched either to manual mode (roughly 30 %) or the original factory settings are still in use (25%). And a great deal of the controllers in automatic operation (30%) actually increases the undesirable variability of the controlled variables. Many surveys have been published and different figures are presented by different authors but the message is clear – the great majority of control structures applied in the process industry should be tuned or redesigned completely [28, 62]. In some cases, the practicing control engineer might be responsible for several hundred control loops [2] and this naturally shows up as a lack of maintenance. In some cases, the reason can be lack of knowledge ("Tuning of the controllers would have barely any effect on the performance!") or old conventions ("That controller has always been on manual!").

Today, systems engineering deals with larger and larger models. For example, in *system dynamics* the analyses of business strategies, logistics or production processes typically result in complex nonlinear dynamic models [64]. Within the process industry the plant model oriented design practices are the hype of the day but many years are still needed until they will reach the status of a standard working practice in reality. Despite the number of reported success stories in the past [7, 81], simulation is still sometimes considered to be a laborious side-track alongside the ordinary assignments, causing only extra effort instead of being an everyday engineering instrument facilitating the design work. Perhaps one reason for the opposition is the lack of practical tools to assist the parameter optimization that is generally considered a tedious and laborious task. Engineers simply hesitate to apply working practices they feel insecure about.

1.2 Scope of the thesis

This thesis concentrates on the restricted complexity controller design problem, i.e., controller tuning, along with model parameter optimization for large-scale systems. It means that problems related to selecting the controller structure using, for example, optimal or robust control theory, are not considered. Respectively, the accuracy improvement of simulation models is pursued only by parameter optimization without considering alternatives for the given structures of the models. Both control and model parameter optimization problems can be seen in a similar framework. A new practical, intuitively appealing, general-purpose working practice both relying on and promoting the use of simulation can be developed in order to solve them.

The optimization of performance via parameter tuning can be seen as a *multivariable multiobjective data-based stochastic nonlinear optimization* problem. First of all, the number of *decision variables* is always high in systems with industrial relevance. Secondly, the problem is numerical since it does not involve a set of equations that could be solved analytically, but rather a *black box* type of cost function that needs to be *sampled* somehow in order to gain insight of its behavior. The problem is *stochastic* since the observations of the cost function contain stochastic variation. Anything general on the global cost function form cannot be argued and it is, therefore, only assumed that the cost function is globally a nonlinear *smooth* function.

1.3 Contribution of the author

This thesis presents theoretical development and practical applications of ideas originally suggested by Prof. Hyötyniemi concerning neocybernetic systems. The loose link between neocybernetic and complex technical systems is exploited to draw ideas and analogies which facilitate solving the underlying research problem – performance optimization of large-scale multiparameter technical systems. Algorithmic development and work concerning the presented case studies has been accomplished by the author under the supervision of Prof. Hyötyniemi with the exception of the RIRT algorithm which was developed by M. Lehtomäki under the tutelage of the author. The modeling of the Apros models applied in the three case studies presented in Sections 5.2, 5.3 and 5.4 was performed by external professionals.

Theoretical and practical problems related to control parameter tuning and model parameter optimization have been studied in this thesis. The most significant and widely

applied methods along with the promising new research results of recent years have been summarized to give a background for the research problem of the thesis. Both problems are formulated into the same mathematical problem framework for which applicable methods are studied and combined into a coherent solution.

A method called Iterative Regression Tuning (IRT) is proposed for the research problem and its applicability is demonstrated in several different case studies. The IRT method represents a new way of handling large complex systems concentrating on statistical examination of the emergent properties of the system. The ideas presented can be applied for developing new working practices and tools for everyday engineering work in the near future.

The software tool called *Tuning Tool* that uses the IRT algorithm was designed and for the most part implemented by the VTT research personnel. The author has been responsible for the Matlab code of the IRT algorithm, for testing of the Tuning Tool and for some improvements and corrections of the Tuning Tool code.

1.4 Structure of the thesis

In Chapter 2 methods for accuracy evaluation of simulation results and different model parameter optimization techniques are surveyed. Correspondingly, a short overview of control performance assessment and controller tuning methods proposed in the literature and their application in practice is given.

Chapter 3 introduces first the fundamental idea of the Iterative Regression Tuning (IRT) method and secondly clarifies the different components of the algorithm in more detail. At the end of the chapter two justifications for the chosen methodology are given. The first one is an intuitive justification grounding in neocybernetics and the second one is based on characterization of the underlying optimization problem.

Chapter 4 outlines the use of the IRT method for control and model parameter tuning in different configurations.

Chapter 5 presents the application of IRT method with several case studies. First, small examples using simple Matlab models are given to demonstrate the research problem after which more realistic examples using professional Apros models are presented both related to control and model parameter tuning problems.

In Chapter 6 the final conclusions concerning the thesis and the presented research results are given.

2 MODELING AND CONTROL OF LARGE-SCALE INDUSTRIAL SYSTEMS

This chapter introduces past and present approaches to model accuracy evaluation and parameter optimization, control performance assessment and controller tuning. Also application of the proposed techniques in practice is discussed.

2.1 Modeling of industrial processes

It is not far from the truth to claim that all engineering work includes modeling in one way or another. Mathematical models form the basis for the analysis and design of technical systems. Solely in the area of the process industry, modeling has numerous applications including process and controller design, production management and optimization, fault diagnosis, soft sensors, model predictive control, training and operator support systems. The models designed for these different purposes concentrate on different aspects of the modeled system and, therefore, using a model for an application other than the original one rarely works out well.

2.1.1 Terminology

The term *model* can be used arbitrarily in many different contexts. Within this thesis, however, model refers to a mathematical representation of a system that captures the essential parts of the system. The end use of the model determines what is meant by the essence, in other words, is it the internal structure and functionality of the system or its observed overall behavior. In practice, model is always a simplified version of the existing system. The simplifications can be intentional, accidental or forced by insufficient knowledge of the system. And once again, it is the intended end use that determines the sufficient level of accuracy. And vice verse, the representativeness of a model dictates its applicability to different uses.

Systems interact with their environment and so does their mathematical counterparts. Systems react to the stimulus or excitation of certain environmental variables and similarly the state of a model is affected by changes of the *input variables* or *input signals*, denoted by u. Respectively, the state of a system or model is reflected to the environment through the set of *output variables* or *output signals*, y. The mathematical model is a function describing this input-output relationship. The model can be either *static* (instantaneous) or *dynamic* (including delays and dependencies from the past input variable values).

Model always consists of a fixed structure and a set of parameters θ. By changing the parameter values (i.e., the *parameterization* of a model) different input-output mapping is obtained. Some parameter values can be fixed with an adequate precision, for example, based on physical laws (e.g., acceleration of gravity) while other parameters without any clear meaning need to be *identified* or *estimated* based on data (e.g., the coefficients of a time series model). Most of the parameters fall between these two extremes. Typically, domain area specialists can specify a probable range for the values but data based inference is needed to find the exact values.

The model itself can be analyzed in order to study the system it describes or it can be used for *prediction* and *simulation*. Usually the term *predict* is reserved for forecasting future output values, $\hat{y}_{\text{future}}$, of a (dynamic) system from a known initial state, y_0, (being a function of the past inputs and states, u_{past} and y_{past}) using the given future inputs, u_{future},

$$\hat{y}_{\text{future}} = g\left(u_{\text{future}}, y_0\left(u_{\text{past}}, y_{\text{past}}\right)\right), \tag{2.1}$$

where the circumflex is used for separating the calculated output estimates from the actual observations. The term *estimate* is commonly used as a synonym for all model outcomes to emphasize the uncertainty of results which is especially true for stochastic systems. The correct determination of the *initial state* or *initial conditions* is as crucial as the correctness of the input signals what it comes to the accuracy of the output estimates. The term simulation is used in a much wider sense quite often referring to any calculations using a model and a set of input values. Typically, simulation is associated with commercial modeling and simulation software that offer also visualization tools and other functionality in addition to mere numerical output estimates. In this thesis, the term simulation is used for the calculation of dynamical system responses to the given stimuli from a given initial state.

In practice, the terminology concerning models and initial conditions is rather inconsistent. In everyday use, *model* typically refers to a combination of structure, parameterization and initial condition of variables of the modeled system.

2.1.2 Model accuracy evaluation and improvement

The inconvenient truth is that models are always wrong (including the environmental models proposing the climate change). However, all models are not completely useless and forecasting future climate conditions is possible. However, the predictions are true only with a certain probability.

The structure of the model, the parameterization and the applied inputs affect the reliability of the output estimates. Construction of realistic simulation scenery with respect to the initial condition and all input signals is challenging especially for models of large-scale systems.

The reliability of output estimates can be characterized (before obtaining the actual output measurements) using confidence intervals or error tolerances if the model structure is simple enough. Realistic models of industrial processes are, however, without exception too complex to be given any analytically calculated guarantee for the accuracy of output estimates. Using *sensitivity analysis* it is possible to characterize how much the

output estimates vary if the set of input signals, parameterization or the initial condition is slightly altered.

If the actual output measurements are available it is easier to evaluate the accuracy of the model outputs by calculating different error measures such as Root Mean Squared Error (RMSE),

$$q_{\mathrm{RMSE}} = \sqrt{\frac{1}{T}\sum_{i=1}^{T} e_i^2} = \sqrt{\frac{1}{T}\sum_{i=1}^{T} (r_i - y_i)^2}\,, \tag{2.2}$$

where y_i is the model output, r_i the corresponding reference signal and T length of the signals. If the square root is omitted from (2.2) the error measure is called Mean Squared Error, MSE. Usually, the main interest is on the mean value of the error signal and, therefore, the random variations are filtered out by approximating the expectation value of the error signal with the arithmetic mean. In some cases, also the error variance can be an important figure. The weighting of certain important periods of the error signal is possible but requires lots of handwork in practice. In some cases, the accuracy of the frequency-response is of greater interest than the time-response. Then the error measure can be computed over the frequencies instead of the time instants.

Based on the accuracy evaluation of the model, its structure or parameterization can be improved. Changing the structure of the model requires gathering of more detailed knowledge of the modeled system, whereas the model parameter values can be improved more easily using data-based techniques. If measurement data from the system is available, it is possible to find the optimal model parameterization analytically, if the system is simple enough (methods of system identification are presented, e.g., in [53]). Usually one has to rely on the more general optimization methods presented in Section 2.3 in order to improve the accuracy of the model.

2.1.3 Industrial practice of modeling and simulation

In the following, the use of modeling and simulation in process industry and some examples of applied software are presented. Also the prevailing practices of model parameter optimization are discussed.

Modeling and simulation of industrial processes

DuPont is one of the world's largest chemical engineering companies. The use of dynamical process modeling and simulation as an everyday engineering tool in the company has been described by Cox *et al.* in [7]. The authors conclude that there are several challenges to be solved before a paradigm shift to model-based, simulation assisted engineering can take place. One of their major concerns is how to increase the number of potential users of dynamical modeling (and especially the already existing models) in an engineering organization. If the models are built to be used only once the modeling costs become intolerable. Cox *et al.* list several future challenges to simulator suppliers and end-users, one of them being implementation of a modeling environment or framework that permits easy parameter identification, sensitivity analysis and other studies. Also Dochain *et al.* conclude that improvements reducing the modeling effort are required since the cost of modeling is the current bottleneck of industrial applications [9].

One approach to attack the above mentioned challenges is presented by Karhela [42] and Kondelin *et al.* in [48]. They propose a specification for a web service oriented framework integrating the process plant life cycle information management, modeling, simulation and other value-added services such as parameter optimization tools. Another example of such a service oriented framework and a case study of parameter estimation in complex environmental models is presented in [30].

Matlab and *Simulink* are nowadays the standard tools for scientific modeling and simulation which are also widely used in industry. Together with the wide selection of specific toolboxes that are available, they offer powerful tools for the mathematical analysis of different systems. Their professional usage for modeling large scale industrial processes is, however, rare since the software does not provide detailed process component libraries. Application specific libraries become necessary when accurate plant wide models are considered.

Modelica is an object-oriented modeling language for a large variety of systems. It offers public domain model component libraries supporting the modeling task and can be applied, for example, to mechanical, electrical, electronic, hydraulic, thermal, control or process system modeling.

Apros (Advanced Process Simulator) is an example of advanced professional modelling and simulation software for large-scale systems of the process industry. Apros was developed by the Technical Research Centre of Finland (VTT) and Fortum Plc (former Imatran Voima) in the 1980s for modeling nuclear and combustion power plants. Since then it has been extended to cover also the processes of the pulp and paper industry. It provides large libraries of different process and automation components, which can be combined into rigorous plant-wide models of industrial processes. Apros has been used successfully, for instance, in training simulator, automation testing, control design, process optimization and safety analysis projects [81]. In most of the case studies presented in Chapter 5 Apros models have been applied. For this reason, Apros has been introduced in more detail in Appendix A.

Aspen Plus Dynamics, *gProms* and *Flowmaster* are examples of other commercial software for process modeling and dynamical simulation.

Model parameter optimization in practice

In practice, model parameters are fitted to data using different tools ranging from large commercial software like Matlab to small parameter tuning toolboxes that are mainly used within a relatively small domain, for example, PEST and UCODE are used for model parameter estimation mainly within environmental modeling [52, 67]. MOSCITO is another example of data based optimization tools designed for modest-sized Micro-Electro-Mechanical systems (MEMS) using either nonlinear LO or heuristic GO methods [46, 72]. One reason for the scatteredness of this area is that different optimization problems rise from the models applied in different fields of science and, therefore, it is often practical to use tailor-made parameter optimization tools.

Regardless of the software that is used for parameter tuning, the number of estimated parameters is typically kept in minimum to the last and as many parameters as possibly are fixed using domain area expertise or less educated guesses. Only the parameters lacking any interpretability are estimated from data and usually in several batches (one model or process component, or one parameter, at a time). The reluctance to use

numerical parameter optimization as a standard approach seems to originate largely from the practical inconveniences related to the task, such as interfacing the model with optimization packages.

There exist also several general purpose optimization packages that are not restricted to any specific application domain. SNOPT is one of the most renowned commercial linear/nonlinear constrained optimization tools [10]. It is applied, for example, within Comsol Multiphysics simulation software. SNOPT is an implementation of a particular Sequential Quadratic Programming (SQP) algorithm. In SQP the original nonlinearly constrained problem is transformed into a sequence of linearly constrained subproblems in which the Lagrangian function is replaced with quadratic approximations. It is assumed that the first derivatives are available which restricts the application of SNOPT to some extent.

NIMBUS (Nondifferentiable interactive multiobjective bundle-based optimization system) is another general purpose optimization tool which has been designed for both differentiable and nondifferentiable multiobjective and single objective optimization problems subject to nonlinear and linear constraints for the variables [60]. It is primarily designed for analytical equation based optimization but also applicable to problems without analytic target function formulations. NIMBUS emphasizes the importance of usability in decision making offering user several possibilities for scalarization of multiple targets.

The above mentioned examples are only excerpts from the abundance of the available software packages for the purpose. Still, it is hard to point out a widely applied specific tool for parameter optimization of dynamic process models. Matlab provides powerful algorithms for parameter optimization but the customary way to handle individual model blocks at a time does not exploit its full potential. The IRT method presented in Chapter 3 tries to fill this gap.

2.2 Control systems in process industry

One of the current key challenges of the process industry, pointed out by Dochain *et al.* [9], is the increasing global competition. Modern plants located in the developing countries are producing comparable high quality products into market at competitive prices. To maintain the competitiveness of older process plants in, for example, Northern Europe, that are suffering from the increasing energy, raw material and labor costs, the profitability and the efficiency of the production need to be increased in every possible way. Completely new process technologies and innovative solutions, like biodiesel production from woody biomass, are one resort for the mature process industry. In order to maintain the traditional operation of industry, profitability needs to be enhanced by improving the performance of the existing process facilities.

It is a well-known fact that only the most crucial parts of the control systems in process industry are maintained and tuned regularly. Typically, the less important controllers are never tuned unless they are causing notable harm and in many cases the factory settings of parameter values are used even years after installation. However, these controllers of minor importance still have an effect on the overall process performance and their tuning should not be neglected completely. The *control performance assessment* (CPA) and

different *controller tuning methods* have been intensive research topics during the past decades and a variety of methods and techniques have been developed and published.

CPA is usually considered as a subtask within *process monitoring* together with diagnosis and clearing of the faults. When a badly behaving controller has been detected, the reason for the problem needs to be determined before the correcting operation can take place. The poor performance can be caused by, for example, faulty actuator, measurement or other process device, bad process design, inappropriate controller tuning or structure. Typically, controller tuning is easily fixed compared to the other faults mentioned above (if the badly behaving controller is only detected among the hundreds of control loops in the first place). Process monitoring, fault detection and diagnosis fall beyond the scope of this thesis and are not discussed further although they are closely related to CPA and controller tuning in practice. In the following, first, the means of control performance assessment are reviewed, and second, the latest controller tuning methods presented in literature along with the traditional tuning guidelines are introduced.

2.2.1 Control performance assessment (CPA)

The fundamental meaning of *control performance indices* (or measures) is to express the quality of control actions of a controller as a quantitative figure that can be compared with the targets set to the controller in question. Thus the somewhat abstract concept of *overall system performance* can be reduced into a finite set of index values. By comparing the index values to the targets, the performance of countless individual control loops on an industrial plant can be *monitored* significantly easier than, for instance, by examining manually the numerous measurement signals. The monitoring separates out the control loops in need of retuning from the well behaving ones and thus saves the time of plant personnel for other purposes. In the following, only a small proportion of the available CPA methods are introduced. Several thorough reviews of CPA methods have been published over the past 15 years, one of the most recent being presented by Jelali [39].

Many of the control performance indices are developed for SISO controllers, which is understandable since the more sophisticated MIMO control structures are typically so few in number that the control engineers manage to take care of them without any monitoring tools. They are also usually responsible for the most crucial parts of the process and hence their operation is optimized in an appropriate manner anyway. The CPA methods can be divided (roughly) into *deterministic* and *stochastic*, depending on the goals of the controller (tracking vs. regulation problems). In the following section, some traditional textbook characterizations and some more recent CPA methods are presented both for deterministic and stochastic purposes.

Deterministic performance assessment

Traditional time-domain characterizations of control performance in a transient situation are *overshoot*, *rise time*, *settling time* and *decay ratio*. These measures are calculated based on a step response of the system (or after an abrupt load disturbance). Slightly different definitions for these terms can be found from different basic control engineering textbooks, for instance, from [11, 51, 56].

One typical form of bad process performance caused by improper controller tuning is continuous oscillation of the controlled variable. Oscillations cause increased energy and raw material consumption, non-uniform end product quality and unnecessary wear of process components. In [15] Hägglund proposes an *oscillation index* for automatic detection of oscillating control loops. Bad controller tuning is only one possible reason for oscillations, the most typical reason being friction in the control valve and the stick-slip motion it creates. Another reason might be an oscillating load disturbance that may result from another oscillating control loop. In [77] Thornhill and Hägglund propose some methods for the characterization of the detected oscillations.

Hägglund has also proposed an *Idle index* for detecting sluggish control loops [16]. It is rather customary in the process industry to tune controllers conservatively (i.e, excessively slow) in order to avoid instability and oscillations in varying operating points. Consequently, the controllers respond unnecessarily sluggishly to load disturbances and the process needs more time after grade changes and other transient situations to reach the new operation point. This causes long deviations from the setpoint values and, in the worst case, it increases the time of off-spec production.

Stochastic performance assessment

Any statistical figure that is estimated from measurement signals of a plant can be seen as a performance assessment index, for example, variance of controlled variable or mean value of the control error. These measures can be calculated usually also recursively and, therefore, they are easily applied for online diagnostic purposes.

Error signal integrals are general-purpose error measures and they can be used both for deterministic and stochastic characterization of control performance,

$$q = \int_{t_0}^{t_T} (t - t_0)^{\alpha} \left| e(t) \right|^{\beta} dt \,. \tag{2.3}$$

Above, $e(t) = r(t) - y(t)$ is the error signal (difference between setpoint and controlled variable), t is the time variable, t_0 and t_T are the bounds of integration, and the notation $|\cdot|$ stands for taking the absolute value of signal elements. The values of parameters α and β can be chosen rather freely if only $\alpha \geq 0$ and $\beta > 1$. In general, the values of error signal integrals alone do not tell much about the performance unless the values are compared with the history. For example, the integral of absolute value of error (IAE index) is obtained from equation (2.3) with $\alpha = 0$ and $\beta = 1$ whereas the integral of time-weighted squared error (ITSE index) with $\alpha = 1$ and $\beta = 2$. The selection of t_0 and t_T has a bigger influence instead. For example, the settling time can be characterized with the ISE index ($\alpha = 0$ and $\beta = 2$) practically if t_0 is set to the instant of maximum overshoot of a step response. The use of different error signal integrals as target functions within controller tuning is elaborated more in [45]. The formulation of the ISE index coincides with the MSE criteria (2.2) only with the exception that MSE is scaled by the length of the error signal.

In regulatory control the variance of the controlled variable measures how well the stochastic disturbances are compensated by the controller. The values of the variance are, however, unbounded and incommensurable, just as the error signal integrals discussed above. The idea of using the variance achieved with the so-called minimum variance

controller as a benchmark was first proposed by Harris in [26] where the *Minimum variance index*, aka the *Harris index*, was introduced. Since then many improvements to the Harris index and different ways to calculate the index value have been reported, see for instance [32]. The Harris index is perhaps the most widely used (or, at least, the most published about) CPA method even though it has its own weaknesses. For example, the index values lose their meaning if the required estimate of the dead time is substantially wrong. Furthermore, it was developed for assessing regulative control tasks and, therefore, it does not fit well for tracking controllers.

Plant-wide CPA in practice

Traditionally, the control performance assessment and optimization of large industrial processes have been handled hierarchically and individual control loops one after another. Due to this laborious procedure only simple diagnostics have been applied and advanced model based benchmarking techniques have not been implemented largely by the industry [39]. According to Harris *et al.* [28], in 1999 only a minority of the industrial plants utilized any system for reviewing the performance of the controllers relative to their design objectives. In [74] Stanfelj *et al*., for example, present a typical hierarchical performance assessment procedure for SISO controllers using autocorrelation and cross-correlation functions. In [29] an expert system is used to assess the SISO control loops of a large plant using the Harris index. In [14] Generalized Minimum Variance (GMV) benchmarking is applied and the goal is to connect the lower level technical control performance objectives with the higher level plant-wide economic targets, but the analysis (still) begins by breaking down the large-scale problem into manageable proportions. Some CPA methods for multivariable control systems have also been developed. For example, in [11] Harris *et al.* present an extension of the minimum variance index for a multivariable case.

It has only been since the early years of the 21st century that the number of the implemented applications in process industry has been increasing. Some of the reported applications are more or less tailored to the processes under examination, for example, in [47] the CPA approach has been developed particularly for the target petrochemical process. In practice, it is practical to incorporate a good selection of CPA indexes into the assessment task [37]. The CPA software presented in [63] includes also some expert system features that assist novel users to interpret the assessment reports. Other industrial application examples are presented, for instance, in [33, 37]. In [39] Jelali presents the most recent review of CPA methodology, industrial applications and available commercial software tools. In practice nowadays every respectable automation supplier offers some tools for CPA and controller tuning. Despite the growing interest towards CPA, there is still a lot of work to be done on the factory floor level. For example, in [4] Bars *et al*. forecast that, among other things, the design of restricted complexity controllers, particularly in cases of complex systems, will still remain as one large worksite for control engineers in the future.

Actually, the term *process performance assessment* should be used instead of CPA to emphasize the fact that it is the performance of the entire process plant (quantified with, e.g., high-level economical and ecological quality measures) that one is interested in assessing and optimizing, not only some odd technical details related to individual low-level control loops. In fact, the strict mathematical optimum with respect to some control performance indices rarely describes particularly well the true desirable process performance. For example, under minimum variance control the manipulated variable (or

the control signal) is assumed to work very aggressively in a large range which in practice may not be possible or is at least harmful to the actuator. Moreover, even though individual control performance indices are usually more or less parallel to the overall plant level targets they can be sometimes conflicting with each other. In such cases domain area expertise is needed to determine the most desirable compromise. The human expertise and decision making capability, as well as the different control performance indices, should be somehow combined.

2.2.2 Classical controller tuning

Different figures have been proposed by different authors, but about 90 - 98 per cent of the controllers are PI or PID controllers in different fields of the process industry [28, 62]. For example, many higher level multivariable controllers actually provide set point values for the lower level controllers that are typically PID controllers. This makes the basic PID controller an essential part of the modern control systems and their proper tuning is a necessity for the satisfactory performance of the overall system.

Ziegler and Nichols presented their famous tuning principles for PID controllers already in 1942. Soon it became clear than these rules of thumb did not always result in a satisfactory performance. Since then numerous adjustments to the original work has been presented and many totally new tuning principles from different starting points have been proposed. Altogether, a huge number of tuning methods with detailed instructions for selecting the three parameters of PID in different situations have been proposed.

The popularity of the PID algorithm results from its simplicity and intuitiveness. No deeper understanding of dynamic systems is needed to be able to still capture the fundamentals of its functioning and the roles of the three tuning parameters. When alternatives to a PID controller are considered, the same problem rises always: Advanced control algorithms, for example, the general linear controllers, are much more troublesome to design and tune. In practice these advanced controllers always result from a model based design practice in which the process model is tried to invert (as accurately as possible). One simple design paradigm is the *Internal model control* (IMC) design principle (see, e.g., [11]). Also these controllers leave us with a couple of tuning parameters (e.g., the closed-loop time constant in IMC design is chosen “manually”, i.e., *lambda tuning*) that need to be “tuned” somehow in the implementation phase at the latest.

Clearly, lack of tuning methods does not explain the amount of badly tuned controllers in the process industry. Quite the contrary, the abundance of methods scattered among journals, conference papers and books published over several decades could actually be the reason, why the knowledge about the latest research results finds its way into practical applications only at a relatively slow pace [62].

In [62] alone, over 200 PID tuning rules from the past decades for different process models have been listed, and each of them fulfills an optimality criteria! Clearly, different tuning methods fit best for different applications (i.e. for different processes with different control objectives) which makes it laborious to find the correct approach. Furthermore, different manufacturers implement different versions of the controllers which for its part makes the application of the tuning rules less straightforward. So a single general-purpose tuning rule that fits for every circumstance just does not exist.

In the following section, first, a short introduction is given to the techniques that are applied in order to avoid the manual tuning of controllers, namely *auto-tuning* and *adaptive controllers*. Secondly, a few more general controller tuning and performance optimization methods are presented. These methods share the same "less rigorous" approach that does not rely on rigorous analysis of system model but uses iterative, numerical, data-based methods instead.

Automatic tuning techniques

Automatic tuning techniques, the so-called *auto-tuners*, raised the interest of academic and industrial people during the 1970s and 1980s. Automatic tuning refers to methods for automatic control parameter tuning *on the operator's initiative*. One state-of-art review of relay feedback auto-tuning is presented in [24].

One of the first commercialized relay feedback auto-tuning methods was proposed by Åström and Hägglund in 1984. The tuning of SISO controllers is based on estimation of the process frequency response at the critical frequency and applying (originally) the Ziegler-Nichols tuning principles. Even though auto-tuners ease the tuning of numerous controllers in a process plant, they still do not work well autonomously. The tuning method, i.e., the target for performance, need to be set manually for each controller. Also the process model, for example, first order plus time delay, needs to be chosen before the relay identification. And furthermore, the method also causes the extra perturbation to the process as the controller is temporarily overridden with the relay for the identification purposes.

Several other auto-tuning techniques have been proposed in the literature [25]. For example, Hang and Sin developed a cross-correlation based auto-tuner, and Bristol proposed a pattern recognition of error signal based auto-tuner. In order to invoke the dynamics of the processes, all these methods require certain excitation of the input signals, which naturally disturbs the production.

Adaptive control

Adaptive control has been developed largely for nonlinear systems, such as aircraft control systems, which cannot be controlled satisfactorily over a wide range of operating points with constant control parameters. Later on, adaptive control has been adopted also by the process industry in order to get rid of the constant need for tuning the controllers. The concept of adaptive control differs from the auto-tuners such that the control parameters are tuned automatically online without any operator intervention. An adaptive controller modifies its behavior in response to changes in the process dynamics and disturbance characteristics, i.e., it adapts to the environment. In principle, the adaptation is based on the identification of process model (parameters of a certain process model) and solving the best (restricted complexity) controller to optimize the chosen design criteria.

Gain scheduling is the most primitive form of adaptive control in which the control parameters are changed as a function of the operating point. This simple system can be implemented, for example, by using a lookup table into which suitable parameter values corresponding to different operating points are stored beforehand.

In *Model-Reference Adaptive Control* (MRAC) the desirable system response to any input signal is calculated using a reference model that specifies the performance

requirements. The difference from the target performance is then minimized via parameter adaptation using, for instance, the MIT rule. Another type of adaptive controller is the *Self-Tuning Regulator* (STR). In the STR the process parameters are first estimated online after which the controller design problem is solved, i.e., STR is an *indirect* adaptive controller.

The adaptive control suffers from a serious drawback, namely from the problem of *closed-loop identification*. Under feedback control the inputs of a controlled system depend on the previous outputs of the system. This means that the parameters of the process model cannot be determined uniquely if the stimulus in input signal does not remain sufficiently large [3]. As a result, the adaptive controller may become unstable.

2.2.3 Numerical controller tuning

After the enthusiasm for CPA methods during the past decades, many different restricted complexity controller design techniques, i.e., controller tuning methods, have been proposed in the literature covering the subject. Many of these approaches rely on numerical or data-based analysis instead of rigorous classical model-based controller design. Some of the most promising ideas are briefly reviewed in the following section.

Iterative Feedback Tuning (IFT)

In the IFT method, originally proposed by Hjalmarsson *et al*. in [31], a set of control parameters, θ^*, are sought that minimizes the Linear Quadratic Gaussian (LQG) type of design criterion,

$$J(\theta) = \mathrm{E}\left\{\left(G_{\mathrm{F},e}e(\theta)\right)^2\right\} + \alpha\mathrm{E}\left\{\left(G_{\mathrm{F},u}u(\theta)\right)^2\right\}, \tag{2.4}$$

where e is the control error signal, u is the control signal, α is a scalar coefficient and $G_{\mathrm{F},e}$ and $G_{\mathrm{F},u}$ are appropriate weighting filters for e and u. The optimization problem cannot be solved analytically and an iterative gradient descend approach needs to be applied. The method disturbs the process with special *gradient experiments* which, however, do not require large setpoint changes or open-loop tests. The IFT results in good step response and disturbance attenuation as compared to traditional Ziegler-Nichols, IMC or ISE tuning principles [50].

The main problem with IFT is the applicability to MIMO systems. It neglects the interactions between individual control loops (even inside a MIMO controller) since each input-output pair in a MIMO controller needs to be tuned separately which results in a huge amount of gradient experiments for a large system. One needs $n \times m$ process experiments in one iteration step if a MIMO controller with n control signals and m measured outputs is considered. Clearly, the curse of dimensionality steps in and evidently impedes the use of IFT in plant-wide cases.

Virtual Reference Feedback Tuning (VRFT)

Campi *et al.* [6] proposed another interesting data-based controller tuning technique in which any representative set of input-output data $\{u(t),y(t)\}$ from the controlled process can be used in tuning after appropriate filtering, i.e., no process experiments or open-loop tests are required. Instead, a transfer function representation of the desirable closed-loop

system, $G_{CL}(q^{-1})$, (i.e., process controlled with an optimal controller) needs to be specified. The explicit model of the controlled process is not required though. First, the *virtual reference* signal $r(t)$ that would result in the observed response $y(t)$, if applied in the desirable closed-loop system $G_{CL}(q^{-1})$, is computed. Under these circumstances the control signal computed by the optimal controller necessarily coincides with $u(t)$. Hence, the optimal control parameters can be solved by minimizing the design criterion,

$$J(\theta) = \frac{1}{T}\sum_{t=t_1}^{t_T}\left(u(t) - G_C\left(q^{-1}\right)e(t)\right)^2, \tag{2.5}$$

where $e(t) = r(t) - y(t)$, $G_C(q^{-1})$ is the discrete time controller transfer function and T is the length of the time series data.

VRFT is also easily applicable only to SISO systems. Determination of $G_{CL}(q^{-1})$ is not a completely effortless task and designing an appropriate filter for the input-output signals can be laborious, too. Hence, even though theoretically clever VRFT is a slightly impractical procedure to be applied on large industrial systems with dozens of control loops.

Iterative Learning Control (ILC)

Iterative Learning Control (or simply Iterative Learning, IL) was initially developed to improve the transient tracking performance of robots and other mechanical systems performing repetitive operations. Since then it has been applied, for instance, to improve the performance of different batch processes in the chemical, pharmaceutical and biochemical industry. The basic idea is that one tries to find an input signal (or profile) $u(t)$ that minimizes the difference between the reference signal and controlled variable. The method is iterative and the applied input signal is updated between the trials using a learning filter $G_F(q^{-1})$ such that,

$$u_K(t) = u_{K-1}(t) + G_F\left(q^{-1}\right)e_{K-1}(t). \tag{2.6}$$

(Notice that the subscript refers here to the K^{th} iteration instead of the K^{th} element of a vector.) An extensive survey of ILC algorithms and applications is given in [5] where, e.g., the filter design is discussed.

Experiments to improve the tracking performance of continuous processes using ILC have been reported too. For example, in [78] ILC has been applied for PID tuning using time domain characteristics, such as peak overshoot, settling time and rise time of a step response, as design criteria. Actually, the term *Repetitive Control* (RC) should be used for situations in which the initial condition of the system varies on different trials. For example, repeated (identical) control actions on a continuous process start in practice from different dynamical states of the process. Convergence of the iterative learning to an optimum in such cases cannot be guaranteed with the theory related to ILC [5]. ILC and RC have been compared, for example, in [54].

Unfalsified control

In *robust controller design* the main target is to determine an upper bound for the plant model uncertainty. Quantification of the biggest possible model uncertainty enables the control engineers to design robust controllers that are guaranteed to work reliably (and

consequently, sometimes overly conservatively). This attempt is, however, doomed to fail since theoretically it is impossible to deduce such an upper bound based on a limited set of observations [70]. What the observations can offer at best is a *lower bound* on the model uncertainty. This is the underlying idea that is elaborated in *unfalsified controller design*.

In unfalsified controller design the behavior of a system portrayed by input-output data is examined instead of its exact internal structure. One tries to find a controller that fulfills the chosen design specifications using the past input-output data in validation, in other words, a controller that is not falsified by the observed data.

In practice the learning procedure means that the controller is re-parameterized whenever the current tuning becomes falsified, i.e., the obtained performance does not fulfill the design criterion. At the same time the remaining set of possible, unfalsified control parameter combinations is decreased. The determination of set-theoric performance specifications and feasible region in decision space is not necessarily straightforward. Some examples of unfalsified control applied to controller tuning are presented in [40, 69].

Extremum seeking (ES)

Extremum seeking was a widely used tool in control applications in the 1950s that experienced a renaissance in 1990s. It is another model-free iterative optimization approach that can be applied to closed-loop controllers. In ES the gradient estimate of cost function is calculated by using appropriate filtering and sinusoidal perturbation of decision variables. A simulated PID tuning example with results comparable to those obtained using the IFT method is presented [45]. The gradient estimate is deduced based on one function evaluation that involves a step response experiment with the system. This may restrict the application of ES in some cases and makes it unreliable in the presence of noise and numerous decision variables. Moreover, ES can be applied only to one controller at a time which does not advocate its usage on large scale industrial systems.

2.3 Model and control parameter optimization

Thorough analysis on optimization is beyond the scope of this thesis where the objective is solely *model and control parameter optimization*. In the following, a general overview of different optimization methods and classification principles of optimization problems are briefly introduced.

The term *optimization* refers to a mathematical problem, in which an *extremum value*, $J^* = J\left(\theta^*\right)$, i.e., either the minimum or the maximum value, of a *target* or *objective function*,

$$J\left(\theta\right): \mathbb{R}^n \to \mathbb{R}\,, \tag{2.7}$$

is searched for over a *feasible region*, $\Re$, which is a subset of the n dimensional *decision space* $\mathbb{R}^n$, $\Re \subseteq \mathbb{R}^n$. $\Re$ contains all possible value combinations of *decision variables*, θ_i, $i = 1 \ldots n$, $\theta = (\theta_1 \ \ldots \ \theta_n)^T$. Optimization problems are formulated (without any loss of

generality) in this thesis such that $J(\theta)$ is called the *cost function* and its values are always minimized, i.e.,

$$\min_{\theta} J(\theta), \text{ s.t. } \theta \in \Re \subseteq \mathbb{R}^n, \tag{2.8}$$

the optimal value combination of decision variables being,

$$\theta^* = \arg\min_{\theta} J(\theta). \tag{2.9}$$

$\Re$ is composed of *constraints* on θ. These constraints can be taken into account explicitly in $J(\theta)$ if a constrained optimization technique is chosen or they can be handled merely as passive statements reducing the size of the search space. Different ways to incorporate constraints in optimization are presented, for example, in [61].

In practice there are often several target functions, J_i', $i = 1 \ldots m$, $J' = (J'_1 \ \ldots \ J_m')^T$, that need to be optimized simultaneously. In *multiobjective optimization* human decision making, i.e., preference information, between the multiple objectives is required [58] which can be implemented utilizing the appropriate *scalarization*. After applying the scalarizing function, g, the problem is transformed back to a single objective optimization problem and the large body of theory related to that is available.

$$\begin{aligned} J &= g\left(J'(\theta)\right), \text{ where} \\ J' &: \mathbb{R}^n \to \mathbb{R}^m \\ g &: \mathbb{R}^m \to \mathbb{R} \\ \Rightarrow J &: \mathbb{R}^n \to \mathbb{R} \end{aligned} \tag{2.10}$$

Optimization problems and algorithms can be classified in many ways. The structure and scale of the system essentially determine the characteristics of the problem and the choice of optimization methods that can be applied successfully. Only for the very simplest model structures are analytical solution methods applicable, whereas in standard engineering problems, target systems are more like "black box" functions. The term *numerical optimization* refers to a nonlinear optimization procedure that is solved numerically using a computer.

Quite often in the literature optimization methods are divided into *linear* and *nonlinear*. If the system is linear in parameters, for example, an AR (autoregressive) model structure, the unknown parameters can be obtained directly using the Least Squares (LS) approach. For linear optimization problems, hold in general that there exists a unique optimum that can be solved analytically without iteration. In practice, dynamic process models and control systems are much more complex in structure and nonlinear optimization is required. In general, no analytical solutions exist for nonlinear optimization problems and *iterative algorithms* are required. The efficiency of iterative algorithms is measured by the number (and the cost) of iterations and *function evaluations* needed for convergence. In model parameter optimization one function evaluation $J(\theta)$ includes a simulation run using the parameters θ and hence the model has a major influence on the overall time needed for the optimization task. Respectively, the dynamics of the controlled system determine the length of the observation period that is required to perceive the total effect of changing the control parameters, no matter whether the response is recorded from the actual system or from a simulator. The

nonlinear target function may possess several local optima that usually can be approximated with a hyperbolic or parabolic function in the neighborhood of the optimum with a reasonable accuracy [61]. The initial values of decision variables affect largely the number of iterations required for convergence to an optimum and more importantly, they determine to which local optimum the algorithm converges.

Further, nonlinear optimization problems can be divided into *convex* and *non-convex* optimization problems based on the convexity of $J(\theta)$ and $\Re$. $J(\theta)$ is said to be a *convex function* on $\Re$ if for any two points $\theta(i)$ and $\theta(j)$, $\{\theta(i), \theta(j)\} \in \Re$, and for any α, $0 < \alpha < 1$,

$$J\left(\alpha\theta(i)+(1-\alpha)\theta(j)\right) \le \alpha J\left(\theta(i)\right)+(1-\alpha)J\left(\theta(j)\right). \tag{2.11}$$

Similarly, $\Re$ is a *convex set* if for any two points, $\theta(i)$ and $\theta(j)$, $\{\theta(i), \theta(j)\} \in \Re$, and for any α, $0 < \alpha < 1$,

$$\alpha\theta(i)+(1-\alpha)\theta(j) \in \Re. \tag{2.12}$$

Loosely speaking, and without going into details, it can be stated that convex problems are in general much easier to solve. For example, convex problems have a unique global optimum which means that *local optimization* (LO) algorithms (that are typically much more efficient) can be applied instead of *global optimization* (GO). The consequences of convexity are discussed in detail, for example, in [57, 66, 78]. While convexity is a property of an optimization problem, the optimization methods can be classified as LO and GO methods referring to the search domain they encompass in the decision space. Occasionally these terms can be misleading since methods, that are local in principle, converge to the global optimum if the problem is convex. And at the same time, convergence of GO methods to the global optimum in finite time especially for large problems cannot always be guaranteed [61]. In the domain area literature it is customary to recommend combining GO approaches with a LO method. Global methods, even though not highly efficient, are able to escape from local minima regardless of the initial values. And LO methods are more efficient in finding the optimum if only a suitable starting point is given. These approaches are called *two-stage* or *two-phase methods*. *Multi-start* is an approach in which LO is started from several initial values that can be obtained with an appropriate GO method.

If randomness is present in the optimization problem either in the values of $J(\theta)$, in the search procedure or in both one speaks about *stochastic optimization* (or s*tochastic search*) as distinct from the *deterministic optimization*. In many real world problems $J(\theta)$ cannot be declared explicitly as a mathematical expression but the system that is under optimization needs to be *sampled* in order to get any information on its state. The term *data-based optimization* is used in this thesis for such black-box optimization tasks. In practice, data-based optimization usually means stochastic optimization as well since measurements necessarily contain some uncertainties.

In general, the parameter optimization problems related to large-scale industrial simulation models and control systems could be classified into the category of *multivariable multiobjective data-based stochastic nonlinear optimization*. Convexity of the problem is rather impossible to analyse or guarantee due to the black box nature of the cost function (even though $\Re$ is typically convex). However, if the target functions of

the optimization are selected carefully and good initial values can be assumed the problem is unlikely *far from being convex*. These issues are discussed in more detail in Section 3.2. Furthermore, the large number of decision variables (being dozens or hundreds) and extremely costly *function evaluations* (requiring a simulation run with the model) are characteristic to the problem.

In the following a brief introduction on different LO and GO techniques is given. First, the deterministic LO methods are introduced followed by their stochastic counterparts. Thereafter, the presentation of the GO methods is divided into exact and heuristic methods. The main emphasis is on methods designed for real valued decision variables since combinatorial and integer optimization problems do not play any essential role in this thesis. In general, it is impossible to argue whether some of the methods are superior to the others since their performance is highly dependent on the application. The purpose here is to highlight certain features of the algorithms that are relevant to the choices made later on this thesis. For more detailed discussion the reader is referred to the domain area literature, for example, [57, 61, 73, 78, 83]. For the sake of brevity, the obvious dependency on decision variables is not shown in the following formulas, i.e., $J(\theta) = J$.

2.3.1 Local optimization

The most primitive (and inefficient) methods in this category are the so-called *direct search methods*, for example, *simplex search* and *Hooke-Jeeves* methods. Their slow convergence and the high number of function evaluations make their application in large-scale problems unreasonable.

Instead, the *gradient-based algorithms* are the most important group of nonlinear local optimization techniques. Here, only the very basic ideas behind these methods are presented although several modifications and improvements on each of them have been proposed. The common target is to solve (2.9) iteratively by updating the values of θ in the *search direction*, d, such that,

$$\theta(K+1) = \theta(K) + \gamma(K) \cdot d(K), \tag{2.13}$$

where K is the iteration step index and γ is a scalar coefficient (the length of the update step if $||d|| = 1$). The search direction in each method is calculated based on the negative gradient direction, $-\nabla J(K)$, rotated and scaled by a matrix M,

$$d(K) = M(K) \cdot \left(-\nabla J(K)\right). \tag{2.14}$$

For a positive definite matrix M, each iteration step is guaranteed to decrease the cost function value if only γ is chosen appropriately and a deterministic J is assumed. The selection of M separates the different gradient-based algorithms from each other. The value of γ is usually chosen using a *line search method*. Line search algorithms are one dimensional optimization problems that attempt to find the optimal step length in the search direction that maximizes the effect of the update step minimizing J. The search consists of two stages. First, the interval containing the optimal γ is located and after that the interval is reduced so that the required accuracy for the solution is reached. The best known interval reduction methods are the *Fibonacci search* and *Golden Section search* methods. The optimal length of the update step can be solved analytically if the shape of J can be approximated, for example, with a hyperparabola. Such approximation is often

valid in the vicinity of the optimum. How to detect the vicinity of an optimum, however, is another question.

The *steepest descent* (or *gradient descent*) algorithm is the simplest of the gradient-based optimization methods in which $d = -\nabla J$, i.e., $M = I$,

$$\begin{aligned} d(K) &= -\nabla J(K) \\ \Rightarrow \theta(K+1) &= \theta(K) - \gamma(K) \cdot \nabla J(K). \end{aligned} \tag{2.15}$$

Steepest descent method involves no information from the previous steps nor higher order derivatives. Although attractive because of its simplicity, the steepest descent method suffers from the so-called zigzag effect which means that the solution starts to fluctuate heavily around the "correct" update direction if the eigenvalue spread of the cost function Hessian is large (meaning that slopes differ largely in different directions). This effect is emphasized in case of a rigorous line search since then the consecutive update steps become close to orthogonal which eventually shortens the update step and dramatically slows down the speed of the optimization. The *backpropagation* algorithm invented in the neural networks community is essentially a steepest descent algorithm.

In the *Newton's method* second order derivative information is applied to modify the search direction. Matrix $M(K)$ in (2.14) is chosen as the inverse of the *Hessian*, $M(K) = H^{-1}(K)$, i.e.,

$$\begin{aligned} d(K) &= H^{-1}(K) \cdot \left(-\nabla J(K)\right) \\ \Rightarrow \theta(K+1) &= \theta(K) - \gamma(K) \cdot H^{-1}(K) \cdot \left(\nabla J(K)\right), \\ &\text{where } H(K) = \frac{\partial^2 J}{\partial \theta^2}. \end{aligned} \tag{2.16}$$

In practice, this is applicable only if the Hessian, or at least the gradient information, is available in analytical form. In some versions of the algorithm $H(K)$ is replaced by a matrix that is close to Hessian but guaranteed to be positive definite. The so-called *Quasi-Newton method*, also known as the variable metric method, avoids the numerically demanding inversion of H by substituting an approximation for H^{-1}.

In the *conjugate gradient methods* second order derivatives are not needed but information from the previous step is utilized such that the consecutive update steps $\{d(0), d(1), \ldots, d(K)\}$ are said to be *conjugate* with respect to a symmetric positive definite matrix A, i.e.,

$$d(K)^T A d(K-1) = 0, \tag{2.17}$$

which means that for a quadratic cost function $J = \theta^T A \theta$ convergence to the optimum is guaranteed in at most n steps. The update principle becomes,

$$\begin{aligned} d(K) &= -\nabla J(K) + \gamma(K) \cdot d(K-1) \\ \Rightarrow \theta(K+1) &= \theta(K) - \nabla J(K) + \gamma(K) \cdot d(K-1). \end{aligned} \tag{2.18}$$

Determination of γ differs between different conjugate gradient methods. In data-based optimization A is, however, unknown and the shape of J is only approximately quadratic. In general, conjugate gradient methods are much faster than, for example, the steepest descent method and they usually work well even with large n. However, they require a rigorous line search in order to work well and the algorithms need to be restarted every n^{th} iteration.

The *Gauss-Newton* and *Levenberg-Marquardt methods* are famous algorithms extending the ideas presented above to more general cost function forms. The underlying assumption is, however, that at least gradient information is available in analytical form and therefore these methods are badly suited for data-based optimization.

2.3.2 Stochastic local optimization

Local optimization methods presented in Section 2.3.1 assume that the cost function and its derivatives are available in analytical form without any uncertainty. However, in many cases the target function is unknown and data-based optimization using noisy observations is applied.

The stochastic counterpart of the deterministic gradient descent algorithm is the *stochastic gradient method* [68] which is the special case of *stochastic approximation*. The purpose of the method is to find the optimal solution, $\theta = \theta^*$, of a (scalar valued) target function, $J(\theta)$, which is not directly observable. Also, the exact value of $\nabla J(K)$ is assumed unknown. The algorithm searches for the optimum with a similar iteration as the gradient descent method in equation (2.15) with the exception that now the gradient information contains uncertainty,

$$\theta(K+1) = \theta(K) - \gamma(K)\nabla\hat{J}(K). \tag{2.19}$$

Above, $\gamma(1)$, …, $\gamma(K)$ is a gain sequence of positive scalar coefficients and $\nabla\hat{J}(K)$, $\dim\{\nabla\hat{J}(K)\} = n\times 1$, is the unbiased gradient estimate for $J(K)$, i.e., $\mathrm{E}\{\nabla\hat{J}\} = \nabla J$. Despite the noisy gradient estimates the iteration (2.19) converges to the optimum if the gain sequence is chosen properly (see [73]), for example,

$$\gamma(K) = \frac{\gamma_0}{(K+1)^{\alpha}}, \tag{2.20}$$

where γ_0 and α are strictly positive. However, using (2.20) slows down the iteration (2.19) too much in practice and constant value for γ is often used (even though convergence cannot be guaranteed in that case).

The assumption of unbiased gradient estimates is, however, often too optimistic. Finite Difference Stochastic Approximation (FDSA) and Simultaneous Perturbation Stochastic Approximation (SPSA) methods can be used to compute the gradient estimates based on noisy cost function observations [73]. In FDSA the gradient estimate is computed either as a one-sided or two-sided approximation. For example, the two-sided gradient estimate is,

$$\nabla\hat{J}\left(\theta(K)\right)=\begin{bmatrix}\dfrac{J\left(\theta(K)+\gamma(K)\delta_1\right)-J\left(\theta(K)-\gamma(K)\delta_1\right)}{2\gamma(K)}\\ \vdots\\ \dfrac{J\left(\theta(K)+\gamma(K)\delta_n\right)-J\left(\theta(K)-\gamma(K)\delta_n\right)}{2\gamma(K)}\end{bmatrix}, \qquad (2.21)$$

where the i^{th} element of δ_i is 1 and rest of the elements are 0, $\dim\{\delta_i\} = n\times 1$. Two-sided approximation requires $2n$ and the one-sided approximation $n + 1$ samples. In SPSA the gradient estimate is computed based on two cost function observations only,

$$\nabla\hat{J}\left(\theta(K)\right)=\begin{bmatrix}\dfrac{J\left(\theta(K)+\gamma(K)a(K)\right)-J\left(\theta(K)-\gamma(K)a(K)\right)}{2\gamma(K)}\\ \vdots\\ \dfrac{J\left(\theta(K)+\gamma(K)a(K)\right)-J\left(\theta(K)-\gamma(K)a(K)\right)}{2\gamma(K)}\end{bmatrix}, \qquad (2.22)$$

where $a(K)$ is a random perturbation vector, $\dim\{a(K)\} = n\times 1$. The elements of a are distributed symmetrically around zero and they are mutually independent. For example, the Bernoulli distribution can be used for a_i. Because of the larger sample size, the FDSA estimate for the gradient is often more accurate. However, if the efficiency of the algorithms during the whole optimization procedure is compared (using the number of function evaluations as a measure of efficiency) SPSA surpasses FDSA, especially if n is large.

In practice the presented gradient estimation techniques can be combined also with the more advanced local optimization methods, like Gauss-Newton and Levenberg-Marquardt methods. However, using higher order derivative information in data-based optimization is slightly questionable since the estimation of second order derivatives would be based on estimates of first order derivatives [61]. At least, these estimates should be based on abundant sample sizes.

2.3.3 Exact global optimization

The following methods are considered exact since they are guaranteed to convergence to the global optimum (however, only under certain circumstances and perhaps only as the iteration index approaches infinity, and so on). The *stochastic search* methods constitute one important group of these algorithms.

The most primitive GO methods are called *exhaustive search* methods (or naïve approaches). The disparaging names come from the fact that these methods are only applicable to small problems with relatively few decision variables and a well defined feasible region since they suffer severely from the curse of dimensionality (see below). According to Pintér [66] solving problems with $n \geq 5$ using these approaches is hopeless in practice. The most straightforward exhaustive search method is to evaluate J in every point of $\Re$. This is naturally possible only if $\Re$ is finite, and θ_i, $1 < i < n$, are discrete valued variables.

The corresponding approach for problems with real valued θ_i is the *Grid search* where $\Re$ is discretized using an equally spaced grid and J is evaluated at each intersection point of the grid. If J satisfies the so-called *Lipshitz condition*,

$$\left|J\left(\theta(i)\right)-J\left(\theta(j)\right)\right| \le L\left\|\theta(i)-\theta(j)\right\|, \tag{2.23}$$

where L is the Lipshitz constant and $\theta(\iota) \in \Re$, $\iota = i, j$, J is said to be a *Lipshitz continuous function*. The expression (2.23) is a smoothness condition stronger than the regular continuity determining the upper bound for the derivative of J in $\Re$. For a Lipshitz continuous function, it can be shown that for an n-dimensional hyperrectangular feasible region with maximum edge length D the number of grid points, k, is approximately,

$$k = \left(\frac{LD}{e}\right)^n, \tag{2.24}$$

where $e = \|J^* - J\|$ is the desired accuracy for the solution. Thus, the number of function evaluations to obtain a solution with the accuracy e grows exponentially with the dimension n. This is called the *curse of dimensionality*.

In *pure random search* (PRS) the search of optimum is solely based on function evaluations. Actually PRS is a sampling method with a *sampling domain* $\Re_S$ covering the whole feasible region, $\Re_S = \Re$. Samples are drawn from a probability distribution that is typically a uniform distribution. The best observation of cost function, $\tilde{J}$, and the corresponding $\tilde{\theta}$ are recorded and every new sample is compared to them until a good enough solution is found. Convergence of PRS to the global optimum with accuracy e cannot be guaranteed with any finite number of function evaluations, but it can be shown that the method still converges to the global optimum with probability one if the number of samples grows to infinity. Similar to the grid search, the required calculation time grows exponentially with n.

Pure adaptive search (PAS) is another stochastic search method in which the sampling domain is adapted according to the obtained values of $J(K)$ that are assumed to be deterministic. It begins similarly as PRS with $\Re_S = \Re$. After each new set of observations $\Re_S$ is truncated to cover only the *improvement region*, i.e., the level set,

$$\Re_S\left(K+1\right) = \left\{\theta : \theta \in \Re_S\left(K\right) \wedge J\left(\theta\right) \le \tilde{J}\right\}. \tag{2.25}$$

The implementation of the PAS algorithm is extremely difficult since the shape of the set (2.25) possesses an arbitrarily evolving shape. In 2003 Zabinsky stated that there actually exist no direct implementations of the algorithm. Some approximation methods exist however, see [83]. The attractive point in PAS is that the expected number of iterations is linear in dimension, not exponential. PRS and PAS are the two extremes of stochastic search methods and the practical algorithms lie somewhere between these two special cases.

Sequential stochastic search methods consist of the following general framework: An initial point, $\theta(0) \in \Re$, is chosen and the algorithm parameters are initialized. A (set of) candidate point(s) from $\Re$ is generated, J is evaluated and the current point $\theta(K)$ is updated to $\theta(K+1)$ based on the candidate point(s) and the corresponding cost function

value(s). New candidate points are generated until a stopping criterion is met. The two main components of a sequential algorithm are the generation principle of the candidate points and the update procedure. Several stochastic search methods of this type have been reported in literature, some of them belonging more to the categories of stochastic LO methods or heuristic GO methods (see Sections 2.3.2 and 2.3.4).

2.3.4 Heuristic global optimization

The title collects together a vast collection of optimization methods, many of them being inspired by different phenomena familiar from physics or biological systems. Quite often the term heuristic is used in a negative sense arguing, for example, that there is actually no evidence that evolution is an optimal procedure. For example, there is no proof that the development from anthropoid to human happened somehow optimally with respect to speed or end result. Even though heuristic GO methods do include features that prevent them from getting stuck on the nearest local optimum the *convergence to global optimum cannot be guaranteed*. However, they have been reported to work well, for example, on certain combinatorial problems [66]. In many applications the user will be satisfied if only a practical solution meeting the design specifications can be found. The mathematical optimality of the solution is often of secondary importance.

One heuristic method to escape from local optima is to add noise on the parameter updates and hope for the best. Another intuitively more appealing method is the classical sequential stochastic search method *simulated annealing* (SA) that mimics the annealing process of metals (and hence uses somewhat exceptional terminology for the parameters of the algorithm, e.g., *cooling schedule*). In principle, the algorithm proceeds similarly as the general framework above with the exception that non-improving update steps are accepted as well (with a slowly decreasing probability during the search procedure). The motivation for this is that the algorithm thus could escape from the local optima.

The development of *evolutionary algorithms* consisted of two parallel lines of development, one in Europe (mainly in Germany) studying *evolution strategies* and the other in USA considering *genetic algorithms*. In both areas, the underlying idea is about the same – to imitate the evolution process of nature on an algorithmic level. Differences lie mainly in terminology and on how the variables are encoded in the algorithm. Without questioning evolution itself it is still reasonable to ask whether it is doing its job optimally. It is hard to analyze the performance of these algorithms since they involve several parameters by themselves and determining their values affects the convergence speed. There is always a trade off between fast convergence and ensuring a globally optimal solution. However, successful applications are presented, for example, in [30] genetic algorithms have been applied to parameter optimization of large environmental models.

Tabu search is another stochastic search method having some heuristic traits. It is a memory based method that keeps track of the search history trying to avoid visiting the same areas of $\Re$ many times. The most obvious implementation is obtained by using a Tabu list that contains the recently visited points of the decision space. It is best suited to combinatorial problems with integer decision variables, such as the travelling salesman problem, but can be used also for problems with real decision variables. Due to its heuristic nature, many references classify it as heuristic local search method. Nevertheless, in recent years Tabu search has been quite popular approach in large scale

combinatorial problems. *Branch and bound* is another search technique for combinatorial optimization.

3 ITERATIVE REGRESSION TUNING (IRT)

Despite the large number of model parameter optimization and controller tuning techniques presented in the previous chapter, another method, *Iterative Regression Tuning* (IRT), for solving both problems is proposed. The same ideas as applied by Hyötyniemi within the study of *cybernetic systems* (see [35, 36]) are used here to overcome the complexity of large-scale technical systems. It seems that replacing the conventional reductionist bottom-up analysis methodology with a top-down data based approach makes it possible to grasp the problem of dimensional complexity in systems. The change of viewpoint lifts also the design and optimization tasks onto a higher abstraction level opening up new possibilities for everyday engineering.

This method concentrates on the *restricted complexity controller design* problem which means that the structure of the controller is assumed fixed and it is only the control parameters that are optimized. Similarly, the model performance improvements are pursued only via parameter optimization, not by improving the chosen model structure.

The name Iterative Regression Tuning stems from the iterative nature of the optimization procedure in which the local linear approximation of the cost function is iteratively repeated at different points of the decision space. The word tuning refers to the practical controller tuning that is carried out in process plants.

In the following, a brief introduction to IRT (in its simplest form) is given in Section 3.1 and the elaboration of the algorithm is deepened in Section 3.2. The theoretical and intuitive foundations of IRT and comparisons to other methods are presented finally in Sections 3.3 and 3.4.

3.1 Overview of IRT

Both model and control parameter tuning problems can be expressed in a similar optimization framework,

$$\begin{aligned} \theta^* &= \arg\min_{\theta} J(\theta) = \arg\min_{\theta} g\left(q(\theta)\right) \\ &\text{s.t. } \theta \in \mathfrak{R} \subseteq \mathbb{R}^n, \text{ where} \\ & q : \mathbb{R}^n \to \mathbb{R}^m \\ & g : \mathbb{R}^m \to \mathbb{R} \\ & J : \mathbb{R}^n \to \mathbb{R}. \end{aligned} \tag{3.1}$$

The goal is to find the optimal values of decision variables, θ^*, belonging to the feasible region, $\Re$, which minimize the cost function, $J(\theta)$, that is a scalarization over multiple *quality measures*, $J = g(q)$, $q = (q_1 \ \ldots \ q_m)^T$. In Section 2.3 the problem was already classified as a *multivariable multiobjective data-based stochastic nonlinear optimization* problem. The practical restrictions and necessary assumptions characterizing the situation even more precisely are presented next and discussed in more detail in the following sections.

Any dynamic (or static) system with multiple input and output variables, $x = (x_1 \ x_2 \ \ldots)^T$ and $y = (y_1 \ y_2 \ \ldots)^T$, respectively, can be described simplistically as,

$$y(t) = G(x(t), \theta) + e(t), \tag{3.2}$$

where θ are the parameters of the system and e stands for the stochastic variations in y. The function G determines the structure of the system (in this context it is either the dynamic simulation model or the control system) and θ corresponds to model or control parameters, respectively. The performance of the system can be evaluated by examining the values of the output signals with respect to the applied inputs (see Figure 1). Evaluating the performance of a simulation model is straightforward when measurement data from the existing system can be used as a reference for the simulation results. The quality measures can be formulated as presented in Section 3.2.1. For control systems, domain area expertise is utilized to determine the evaluation criteria – what is a good manifestation for the overall control performance? Usually, concepts like robustness and accuracy are pursued and the CPA methods presented in Section 2.2.1 can be used as quality measures q for giving the final mathematical formulation for the goals (see Section 3.2.1 for more on quality measures).

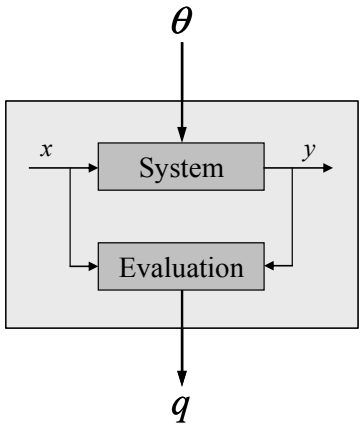

Figure 1. The statistical dependency between parameters, θ, and quality measures, q, describes the relevant properties of the underlying dynamic MIMO system with inputs x and outputs y [35].

The values of m quality measures, $q = (q_1 \ q_2 \ \ldots \ q_m)^T$, corresponding to a certain value combination of n parameters, $\theta = (\theta_1 \ \theta_2 \ \ldots \ \theta_n)^T$, and other variables determining the operating regime, are calculated from a set of input and output signals, x and y. If the system is a simulation model, the evaluation of performance is based on simulated output signals, y, that are obtained when measured input signals, x, from the existing system are used as inputs. In the case of a control system, x and y correspond to control signals and controlled variables, respectively. Since y are affected by random variations, q are also

random variables. However, it typically turns out that there exists a statistical dependency between θ and q, which can be modeled if enough data is available. Evaluation of quality measures based on time series signals is discussed more in Section 3.2.1.

The underlying target functions, q_i, are assumed smooth functions of θ with continuous derivatives which ensures the applicability of gradient based LO approaches. Since q_i are user-defined, the *avoidance of non-convexity* can be sought by selecting appropriate mathematical formulations for q_i, although strict convexity cannot be guaranteed. While J and $dJ/d\theta$ are not analytically available and the observations of q are corrupted by noise, the optimization can still be based on their local model-based approximations.

Due to the data-based optimization approach, a *sampling method* for θ needs to be specified (compare this to *experiment design*). Sampling of θ is restricted to a *local sampling domain*, $\mathfrak{R}_S$, since the feasible region, $\mathfrak{R}$, (i.e., the set of all realistic combinations of decision variable values resulting in a stable solution) in the decision space is generally unknown and cannot be determined exactly beforehand. Furthermore, in case of a process simulator the numerical stability depends not only on the absolute value of θ but also on the size of change in their values. Large abrupt changes may cause the divergence of the solver algorithm. To be able to proceed it needs to be assumed that $\mathfrak{R}_S$ and $\mathfrak{R}$ can be determined such that $\mathfrak{R}_S \subseteq \mathfrak{R} \subseteq \mathbb{R}^n$. Additionally, $\mathfrak{R}_S$ and $\mathfrak{R}$ (being user-defined) can be assumed convex. Sampling is discussed more in Section 3.2.2.

In spite of the underlying details, sampling produces k parameter value combinations,

$$\underset{k\times n}{\Theta} = \begin{pmatrix} \theta(1) & \theta(2) & \cdots & \theta(k) \end{pmatrix}^T. \tag{3.3}$$

After that, the performance of the system with different values $\theta(i)$ is recorded, i.e., either the control performance or the accuracy of the simulation model is evaluated based on the corresponding time series signals resulting in a set of quality measure value combinations,

$$\underset{k\times m}{Q} = \begin{pmatrix} q(1) & q(2) & \cdots & q(k) \end{pmatrix}^T. \tag{3.4}$$

Based on observed data the statistical dependency between q and θ can be modeled. Any smooth function can be approximated with a linear model up to an arbitrary accuracy if the analysis is only concentrated on a small enough domain. Thus, considering small deviations, $\Delta\theta = \theta - \overline{\theta}$ and $\Delta q = q - \overline{q}$, around the current *nominal point*, $\overline{\theta}$, and the mean value, $\overline{q}$, of observations, Q, a linear model can be applied to describe the statistical dependency between $\Delta\theta$ and Δq,

$$\begin{aligned} \hat{q}(\theta) &= \overline{q} + \Delta q \\ &= \overline{q} + F^T \Delta\theta \\ &= \overline{q} + F^T \left(\theta - \overline{\theta}\right), \end{aligned} \tag{3.5}$$

where $\hat{q}$ is an estimate of q and F is a linear mapping F: $\mathbb{R}^n \rightarrow \mathbb{R}^m$. It holds that linear models are an optimal representation of the data in the *maximum likelihood* sense if the

stochastic variations in the observed values of q (i.e. the error terms not explained by the linear model) are normally distributed and θ are deterministic [65]. If q_i are determined according to (2.3) or (2.2) and sufficiently long time series signals are used, it is justified to claim that the above Gaussianity assumption holds according to the *central limit theorem*.

In principle, there is simple way to determine F using the LS estimator (called *Multiple Linear Regression* (MLR) when $m > 1$) which can be computed based on the deviation variables using the pseudoinverse,

$$F = \left(\Delta\Theta^T \Delta\Theta\right)^{-1} \Delta\Theta^T \Delta Q, \tag{3.6}$$

that involves the calculation of the inverse $(\Delta\Theta^T\Delta\Theta)^{-1}$, which is numerically problematic if the dimension of $\Delta\theta$ is high or its components are strongly collinear. The *condition number* (the ratio of largest and smallest singular values of a matrix) of $\Delta\Theta^T\Delta\Theta$ becomes high suggesting numerically inaccurate results, or in the worst case, the inverse cannot be calculated at all. However, *multivariable regression methods* (MVRs) like Partial Least Squares (PLS) offer a solution which might be suboptimal in the maximum likelihood sense but turns out to be superior to MLR in practice and, best of all, makes the approach scalable to multivariable systems. See Section 3.2.3 for more on the local cost function approximation and MVR methods.

The local model F can be applied within LO approaches to determine the search direction in the parameter space that yields better performance of the system. Relatively *good initial values* for θ can be assumed since the reductionist methods presented in Chapter 2 provide a good starting point.

In order to attain a unique solution, the multiple targets of the parameter tuning task need to be *scalarized* into a single cost function J. For example, using a weight vector w results in,

$$J = g(q) = w^T q. \tag{3.7}$$

Decision making between several target functions is discussed more in Section 3.2.4. Despite the seemingly simple optimization target, the problem cannot be solved analytically. Even though the form of (3.7) is linear, the cost function is fundamentally quadratic if only quadratic quality measures are applied. Equation (3.7) can be approximated locally using (3.5) resulting in,

$$\begin{aligned} J &\approx w^T \hat{q} \\ &= w^T \left(\overline{q} + F^T \Delta\theta\right). \end{aligned} \tag{3.8}$$

The gradient of the cost function, ∇J, is obtained by differentiating (3.8) with respect to $\Delta\theta$. ∇J points to the direction of maximal growth and thus, in the basic gradient descent update principle, the values of θ are updated iteratively in the direction of negative ∇J,

$$\begin{aligned}\bar{\theta}(K+1) &= \bar{\theta}(K) + d(K) \\ &= \bar{\theta}(K) - \gamma \cdot \nabla J(K) \\ &= \bar{\theta}(K) - \gamma \cdot Fw,\end{aligned} \tag{3.9}$$

where K is the update step index, d is the update step and γ is a scalar coefficient for adjustment of the step length. The convergence speed of the gradient optimization can be improved in many ways and in Section 3.2.4 the particular LO problem is elaborated in more detail. The IRT procedure contains iteration on two different levels. In the following, the term *local iteration* is used for the successive function evaluations around the nominal point, and the local iteration therefore consists of k local iteration steps. *Global iteration* refers to the iterative update procedure of the nominal point consisting of $K_{\max}$ iteration steps.

3.2 Practical and theoretical details

IRT can be seen as a combination of LO, MVR and stochastic search methods and there exists plenty of freedom in selecting the appropriate components. Application of the algorithm also involves some important technical issues. In the following sections these possibilities and challenges are presented in more details.

3.2.1 Function evaluations

Evaluation of the cost function $J(\theta)$ means that the performance of a system characterized by a collection of time series signals is first compressed into m quality measure values after which computing (3.7) is straightforward. Since IRT can be applied in several configurations, the source of signals and interpretation of quality measures varies. For example, control parameter tuning can be conducted either on an existing process plant or in a simulation environment. Different application possibilities are portrayed in more detail in Chapter 4. To retain the brevity, the discussion concerning control parameter tuning is here restricted to the simulation assisted configuration. In the following, first the selection of appropriate quality measure functions is considered. Second, what is required from the time series signals is discussed.

Quality measures

Quality measures are scalar valued functions expressing how well performance targets are fulfilled. In practice, the target performance is declared by a *reference signal* and the performance by an *output signal* obtained from the system, and it is the difference or error between these two signals that is characterized with $q(\theta)$. In control parameter tuning the user specified reference signal is deterministic, whereas the system output is corrupted by noise (assuming that some random disturbances are added to ensure realistic simulation results). In model parameter tuning the situation is the opposite. The measured reference signal is a random signal realization corrupted by measurement noise. Nevertheless, the source of stochastic variations in signals becomes irrelevant since their difference involves always stochastic variation. Therefore, quality measure definitions should be statistical by nature.

In Sections 2.1.2 and 2.2.1 measures for the accuracy of the simulation results and the control performance were presented. In theory, any of the presented CPA measures could

be used as a quality measure. However, in practice it is beneficial to select such formulations as (2.2) and (2.3) that are statistical characterizations of the lower signal level phenomena. Both equations involve summation over instants of random error signals suggesting that the outputs are approximately normally distributed random variables based on the central limit theorem (which goes nicely together with the local linearity assumption). Using MSE criteria as quality measures is recommended since in that case the cost function becomes a weighted sum of squared errors meaning that its values are necessarily positive, zero being the theoretical minimum value. A good thing about these quality measures is that their application is not tied to any specific situation, for example, to step responses only. The values of q_i should be scaled by the length of the signals so that it has no effect on their expectation values. The commensurability of q_i and q_j is considered in Section 3.2.4.

In practice, it has also turned out that quality measures based on error signal integration are continuous functions of θ, whereas some of the classical CPA characterizations may express severe discontinuities in parameter space and should therefore be avoided in this context. For example, settling time becomes discontinuous in parameter space if it is defined using crisp tolerances around the set point value [17]. Also, overshoot behaves badly as a function of parameters of a discrete time PD controller [78]. Although convexity of a quality measure definition on a local domain around the global optimum (that is hopefully close to the applied initial values) cannot be guaranteed with any quality measure definition, it is important to avoid any nonlinearities in definition of q that could cause discontinuities.

On time series signals and simulated events

The time series signals, based on which q are calculated, should reflect the performance of the system as widely as possible. This means that, for example, in simulation assisted controller tuning the *simulated sequence of events* (or *simulation sequence*) should contain all the relevant events to evaluate system performance. In practice, it is impossible to cover the whole spectrum of possible events from all initial conditions. Therefore, it is usually practical to restrict the evaluation, for instance, only to a certain operation point, production rate or error situation (or to a combination of them), whatever is the main focus of the study. In model parameter tuning the simulation sequence cannot be constructed freely, but a suitable period of data is selected from the available data set. However, the selected period should still be representative of the target system operation and for the future usage of the model. If measurement signals are badly corrupted by noise, filtering of the intended input signals is recommended.

When the IRT algorithm is applied in practice, it should be noticed that after each modification of parameter values, the system is deviated from its dynamical equilibrium. Therefore, periods of time series, that are recorded directly after parameter changes, should not be used for quality measure calculation. The length of the settling time after parameter change strongly depends on the dynamic character of the process.

Finally, the considered time series signals should be sufficiently long, taking into account the noise characteristics and the sampling interval, to ensure reliable performance evaluation with small enough variance of q. Also, the number and nature of simulated events naturally affect the length of the assessment period.

3.2.2 Sampling method

Selection of decision variables in control parameter tuning cases is straightforward since the most reasonable approach is to appoint all control parameters of the target process as decision variables. In model parameter tuning the role and origin of the parameters needs to be considered first. Model parameters can be divided roughly into two categories. Some model parameters, such as characteristic curves of pumps and valves, can be obtained directly from the component manufacturer. These reliable parameters can be excluded from the parameter optimization. Other parameters without as clear interpretation need to be estimated numerically. The division between these two classes is not sharp and lacks any deeper meaning. For some parameters, an experienced professional can assign a range of probable values, for example, flow resistance through process equipment, but the values cannot be determined exactly. It is usually wise to include the parameters from the "grey area" in the set of decision variables as well.

Feasible region $\Re$ and local sampling domain $\Re_S$

The IRT method itself does not require an exactly bounded *feasible region* $\Re$ to be specified, but in practice, it is the easiest way to incorporate the known constraints for θ. In some cases, minimum or maximum values for θ_i are trivial to specify. For control parameters, for example, at least the sign of a controller gain is usually known, even though the upper (or lower) bound is unknown. Similarly, integration time is always positive, and so on. Similar natural constraints for model parameters are easily found, for instance, valve opening is always between 0 and 1. Specifying a minimum and maximum value for each θ_i results in a hyperrectangular $\Re$. It is hard to come up with any situation that would benefit from any more exceptional form of $\Re$. One convenient consequence is that $\Re$ becomes convex which supports (but does not fully justify, however) the use of LO methods.

Data-based LO methods are based on local sampling of the cost function. Therefore, the *local sampling domain* $\Re_S$ needs to be determined. The borders of $\Re_S$ can be determined similarly as for $\Re$ by specifying positive and negative tolerances, ρ_i, for each θ_i around the current value $\overline{\theta}$,

$$\Re_S = \left\{ \theta : \overline{\theta}_i - \rho_i \le \theta_i \le \overline{\theta}_i + \rho_i \forall i \right\}. \tag{3.10}$$

On one hand, local perturbation of θ should be as large as possible, since the variance of (3.6) decreases as the variation of θ grows (see the discussion below about sample size). This naturally speeds up the optimization process. On the other hand, the limits of $\Re_S$ should be small enough to ensure the local linearity assumption and that numerical problems caused by too large parameter changes during simulations can be avoided. If IRT is applied adaptively (see Section 4.2.3) in the target system, keeping $\Re_S$ small enough is even more important since the production process should be disturbed as little as possible. Violations of the local linearity assumption can be detected, for example, with normality testing of residuals (see [76]). In practice, the initial values for limits of $\Re_S$ can be improved during optimization either manually or by using heuristic adaptation rules.

Sampling distributions

IRT uses stochastic sampling methods to avoid laborious deterministic multidimensional experiment design and to ensure good statistical coverage of $\Re_S$. In the sampling procedure, the values of θ are drawn from a *sampling distribution* defined in $\Re_S$. Any probability distribution can be used as long as it is easily parameterized. For example, Gaussian distribution is defined by the mean value $\mathrm{E}\{\theta\}$ and covariance,

$$\begin{aligned}\mathrm{cov}\{\theta\} &= \mathrm{E}\left\{\left(\theta - \mathrm{E}\{\theta\}\right)\left(\theta - \mathrm{E}\{\theta\}\right)^T\right\} \\ &= \begin{bmatrix} \sigma_{\theta_1}^2 & \sigma_{\theta_1}\sigma_{\theta_2} & \cdots & \sigma_{\theta_1}\sigma_{\theta_n} \\ \sigma_{\theta_2}\sigma_{\theta_1} & \sigma_{\theta_2}^2 & & \\ \vdots & & \ddots & \vdots \\ \sigma_{\theta_n}\sigma_{\theta_1} & \sigma_{\theta_n}\sigma_{\theta_2} & \cdots & \sigma_{\theta_n}^2 \end{bmatrix},\end{aligned} \tag{3.11}$$

which determines on its diagonal the variances, $\mathrm{var}\{\theta_i\} = \sigma_{\theta_i}^2$, of θ_i, $i \in [1, n]$ and the covariances between θ_i and θ_j are given by the non-diagonal terms (σ_{θ_i} being the standard deviation of θ_i). In practice, it is initially enough to specify the variances and assume that the non-diagonal terms are zeros. Gaussian distribution is determined between $]-\infty, \infty[$ making its usage on a restricted domain slightly complicated. On one hand, if $\Re_S$ is not explicitly specified there is always a small possibility that a sample far from the mean value is obtained, no matter how small values of $\mathrm{var}\{\theta_i\}$ are chosen. On the other hand, if $\Re_S$ is given Gaussian distribution approximates uniform distribution with large variances (relative to $\Re_S$). Another drawback is that it concentrates the samples around the mean value even though it is more beneficial to draw samples closer to the edges of $\Re_S$. Hypercubical (or hyperspherical) uniform distribution has no other parameters in addition to the minimum and maximum values of θ_i. Therefore, the use of uniform sampling distribution is recommended.

Tabu search modified for real valued θ could also be used to improve the sampling process since it avoids the already visited areas of $\Re_S$. An easy implementation is a memory based *tabu list* approach that contains the already obtained samples. New candidate samples are compared to the tabu list contents and replication of almost identical samples is prevented by discarding candidates when necessary. Comparison of real valued θ requires some kind of a similarity measure or neighborhood function to be defined for θ. For example, *Mahalanobis distance* describes the similarity of two random samples, $\theta(i)$ and $\theta(j)$, taken from the same distribution having the covariance structure R,

$$D\left(\theta(i), \theta(j)\right) = \sqrt{\left(\theta(i) - \theta(j)\right)^T R^{-1} \left(\theta(i) - \theta(j)\right)}. \tag{3.12}$$

In some cases, it is sufficient to compare the angle ω between two samples (if, e.g., the samples are drawn exactly from the boundary of a spherical $\Re_S$),

$$\cos(\omega) = \frac{\theta^T(i)\,\theta(j)}{\left|\theta(i)\right|\left|\theta(j)\right|}. \tag{3.13}$$

Avoiding the heuristic memory-based implementation of tabu search is challenging since constructing a new *a posteriori* sampling distribution after each sample is extremely hard to implement, especially for large n. Similar problems as with the PAS algorithm (see Section 2.3) inevitably arise.

Sample size

Since each function evaluation in IRT involves a computationally expensive time series signal generation with a simulator, it is important to constrain the number of data points to the minimum. It is possible to compute the statistically sufficient sample size for the estimation of LS regression coefficients with a certain confidence. However, such calculations tend to result in impractically large sample sets. In addition, the result may vary a lot depending on the values of the required parameters (like the confidence level, effect size of the regression coefficients, noise variance, etc.) used in calculation. And finally, the result is not directly applicable to PLS regression models.

Since the theoretical approaches to the problem are not altogether free from heuristics, in practice, completely heuristic rules of thumb, for instance, proposing $k \approx 10 \cdot n$, are widely used to "ensure" the statistical reliability of estimates. In both cases, impractically large sample sets are usually proposed from the parameter optimization (of large-scale systems) point of view. The above considerations relate to the estimation of standard LS models (3.6) where (assuming $m = 1$),

$$F \sim N\left(\mathrm{E}\{F\}, \sigma_a^2 \left(\Theta^T \Theta\right)^{-1}\right), \tag{3.14}$$

where σ_a^2 is the variance of the noise corrupting the output observations. From (3.14) it can be seen that the variances of the regression coefficients F are reduced either by decreasing σ_a^2, or increasing k or $\sigma_{\theta_i}^2$ [65] and vice versa. The LS (or MLR) estimator is unbiased as long as the Gaussianity of σ_a^2 holds. In practice, this means longer time series signals or larger (and possibly more risky) $\Re_S$.

Often much fewer samples are needed if *latent variable based multivariate regression* (MVR) methods are applied (see Section 3.2.3). Then, it is not the number of decision variables, n, but the number of *latent variables*, N, that effectively determines adequate k. In some cases, even extremely small sample sizes with $k < n$ can still result in useful regression models. The number of underlying latent variables, being typically only a numerical concept without any clear physical interpretation, however becomes evident only after the model has been estimated. It is an intrinsic quality of the modeling problem and, therefore, selection of k needs in practice to be considered separately in each case. Based on practical experience (see Chapter 5), using sample sizes of $k \geq 2 \cdot n$ is recommended. If the function evaluations (see below) are excessively expensive but data-based optimization is still pursued, relatively small sample sizes just have to be accepted. The number of required samples can be reduced effectively by reusing the old data samples, for example, with exponential smoothing (recursive modeling is discussed in more details in Section 3.2.4).

The affordable sample size also affects the selection of regression method since the best results on an "easy" situation (large k, small σ_a^2 and uncorrelated θ) are obtained with

MLR regression, whereas MVR techniques are superior in the case of insufficient sample size, noisy observations and collinear variables.

3.2.3 Local linear modeling

In the following, estimation of the model (3.5) from a data set is studied. Firstly, the detection of outliers from data sets and reasons for their occurrence are briefly presented. Secondly, the best way to scale and weight variables before modeling is discussed. Thirdly, different regression methods are compared and, finally, recursive modeling approaches are presented to reduce the number of data samples required.

Outliers

Before modeling, the erroneous data points, or *outliers*, should be removed. Modeling is based on Θ and Q, and because Θ is user specified (although composed by random sampling), it can be assumed that it contains no erroneous values and $\theta_i \in \Re_S$, $\forall\, i$. Instead, Q may contain samples that differ significantly from the *joint distribution* of q_i. In simulation assisted controller tuning and in model parameter tuning, numerical problems with the simulator, or other practical problems, may generate abnormal signal values resulting in deviating q values that show up as outliers from the joint distribution. In practice, the occurrence of outliers can be avoided rather effectively by designing the tuning procedure carefully (i.e., by selecting initial values and determining the local sampling domain boundaries sensibly).

If IRT is applied for controller tuning in the adaptive framework (see Section 4.2.2), the observations of q become in general (much) more widely spread due to the rich and varying excitation that a realistic process, with changing operating conditions, possesses and, therefore, the rate of observed "outliers" rises rapidly. In practice, the amount of outliers becomes smaller and the distribution of q values narrower if the length of the time series signals, applied in quality measure evaluation, is long enough and if the optimization procedure is halted during known plant shutdown periods and other exceptional situations.

If it can be assumed that the local linear dependency between q and θ is known and that the random variation in q is Gaussian and independent from θ, the effect of parameter variation can be removed from the observations,

$$\begin{aligned} \mathrm{E} &= Q - \hat{Q} \\ &= Q - \left(\bar{Q} + \Delta\Theta F\right), \end{aligned} \tag{3.15}$$

after which one can concentrate on the distribution of residuals ε, $\mathrm{E} = (\varepsilon(1) \ldots \varepsilon(k))^T$, that should form a zero-mean multivariate normal distribution with covariance R_ε,

$$\varepsilon \sim N_m\left(\mathbf{0}, R_\varepsilon\right). \tag{3.16}$$

Deviating samples $\varepsilon(i)$ that are *improbable* representatives of the same normal distribution (with respect to chosen confidence level) as the rest $\varepsilon(j)$, $i, j \in [1, 2, \ldots, k] \wedge i \neq j$, can be denoted as outliers and discarded from the data set. The main drawback of the presented procedure is that the local model F is needed before it has been estimated

from the data. This discrepancy can be avoided if each sample is left aside at a time and the local model is re-estimated based on the remaining $k-1$ samples. Outlier detection becomes thus an iterative process. This approach is not completely trouble free since in the case of multiple outliers the local model estimate becomes distorted.

There exist several techniques for assessing multivariate normality of a distribution and for detecting multivariate outliers. Both are naturally more complicated than the corresponding univariate problems. In general, non-normality is a much wider concept than normality (compare this to linearity vs. non-linearity) and a distribution can depart from normality in many different ways. Different detection methods emphasize different types of departures and are, therefore, best suited for different purposes.

A simple way to examine the multivariate normality is to assess the normality of *marginal distributions* of ε_i using any procedure for univariate distributions. Marginal normality assures the necessary (but not sufficient) conditions for multivariate normality. A slightly more sophisticated approach is to study the marginal distributions of the principal components z of the residuals,

$$z = \Phi_{\mathrm{E}}^{T}\varepsilon \Leftrightarrow Z = \mathrm{E}\Phi_{\mathrm{E}}. \tag{3.17}$$

The column vectors of $\Phi_{\mathrm{E}} = [\phi_1 \ \ldots \ \phi_m]$ are obtained by computing the eigenvectors of the residual covariance matrix,

$$\frac{1}{k}\mathrm{E}^{T}\mathrm{E}\cdot\phi_i = \lambda_i\phi_i. \tag{3.18}$$

Extensive review of methods for normality testing is beyond the scope of this thesis. For the interested reader, a thorough review by Thode [76] is recommended. It needs to be noted, however, that one should avoid too eager outlier removal since the strongly deviating samples may contain also relevant information, not only disinformation caused by an error. And the reasons behind outliers should be always clarified so that the corresponding errors could be prevented in future.

Data preprocessing

For multiple reasons, it is recommended to apply certain preprocessing of data before the local linear model (3.5) is estimated. First, variables are centered, $\Delta\theta(i) = \theta(i) - \overline{\theta}$ and $\Delta q(i) = q(i) - \overline{q}$. Deterministic decision variables can be centered either with respect to the arithmetic mean of the sample set or to the current nominal point, i.e., the initial values of θ or the results of the previous optimization iteration. $\overline{q}$ is always the arithmetic mean of observations since the observation corresponding to $\overline{\theta}$ is only a single realization of the random variable q. Throughout the thesis, the difference of a *sample mean* and a given mean value is not taken into account when, for example, (co)variances are computed. It is assumed that k is big enough so that $k \approx k-1$.

After centering, the variables should be scaled somehow. First of all, the variables should vary (at least approximately) in the same scale around the mean value to avoid numerical problems. This can be handled by scaling the variances of individual variables to unity,

$$\Delta\Theta = \Delta\Theta_S S_\Theta = \Delta\Theta_S \begin{pmatrix} \sigma_{\theta_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_{\theta_n} \end{pmatrix}$$
$$\Leftrightarrow \Delta\Theta_S = \Delta\Theta S_\Theta^{-1} = \Delta\Theta \begin{pmatrix} \frac{1}{\sigma_{\theta_1}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sigma_{\theta_n}} \end{pmatrix}, \tag{3.19}$$

where the scaling matrix, S_Θ, contains the standard deviations of individual variables on its diagonal. The scaling of ΔQ is performed respectively, $\Delta Q_S = \Delta Q S_Q^{-1}$.

In many cases, it is beneficial to use PLS regression, that tries to detect the directions of input space that correlate maximally with the outputs, in order to reach the most efficient dimension reduction for the input-output model [34]. Now, if either the inputs or the outputs carry an internal correlation structure in themselves (i.e., variables are not linearly independent) and the underlying correlation structure is competing with the input-output cross-correlation, it will distract finding the optimal PLS model. Therefore, whitening both $\Delta\Theta$ and ΔQ *separately* before modeling often results in better PLS regression models than using scaling. The whitening of data matrices also scales variables to unit variance. For whitened data, $\Delta\Theta_W$ and ΔQ_W, the covariance matrix becomes an identity matrix,

$$\begin{aligned} R_{\Theta_W} &= \frac{1}{k}\Delta\Theta_W^T \Delta\Theta_W = I \;\Leftrightarrow \\ \frac{1}{k}\left(W_\Theta^{-1}\right)^T \Delta\Theta^T \Delta\Theta W_\Theta^{-1} &= I \Leftrightarrow \\ R_\Theta &= \frac{1}{k}\Delta\Theta^T \Delta\Theta = W_\Theta^T W_\Theta. \end{aligned} \tag{3.20}$$

From (3.20) can be seen that the whitening matrix, W_Θ, is obtained by the *Cholesky decomposition* of the data covariance matrix, R_Θ, after which whitening is performed similarly as scaling,

$$\Delta\Theta = \Delta\Theta_W W_\Theta \Leftrightarrow \Delta\Theta_W = \Delta\Theta W_\Theta^{-1}. \tag{3.21}$$

Whitening of the quality measure data, Q, is performed correspondingly. Since W_Θ is not diagonal, the matrix (W is an upper triangular matrix) inverse is not obtained as easily as with S_Θ. For a symmetric positive definite matrix (R is necessarily both symmetric and positive definite), W is always real valued.

It needs to be noticed that also the computation of Cholesky decomposition fails similarly as the direct computation of the MLR model from centered data if the smallest eigenvalues of R_Θ are close to zero, i.e., the variables are mutually strongly correlated. However, practical results have shown that when computation of Cholesky decomposition succeeds, whitening has the tendency to improve the numerical properties of the estimated model.

After whitening, $\Delta\Theta_W = \Delta\Theta W_\Theta^{-1}$ and $\Delta Q_W = \Delta Q W_Q^{-1}$, the MLR model between whitened data spaces, F_W, can be estimated from $\Delta\Theta_W$ and ΔQ_W and equation (3.6) becomes simpler,

$$\begin{aligned} F_W &= \left(\Delta\Theta_W^T \Delta\Theta_W\right)^{-1} \Delta\Theta_W^T \Delta Q_W \\ &= \frac{1}{k} \Delta\Theta_W^T \Delta Q_W . \end{aligned} \tag{3.22}$$

The model based representations of the output observations become,

$$\begin{aligned} \Delta\hat{Q} &= \Delta\hat{Q}_W W_Q \\ &= \Delta\Theta_W F_W W_Q \\ &= \Delta\Theta W_\Theta^{-1} F_W W_Q , \end{aligned} \tag{3.23}$$

where the circumflex denotes estimated values. From (3.23) can be seen that F actually equals to,

$$F = W_\Theta^{-1} F_W W_Q , \tag{3.24}$$

and equation (3.5) can be written finally as,

$$\begin{aligned} \hat{q} &= \bar{q} + F^T \Delta\theta \\ &= \bar{q} + W_Q^T F_W^T \left(W_\Theta^{-1}\right)^T \left(\theta - \bar{\theta}\right). \end{aligned} \tag{3.25}$$

About MVR methods

High dimensional data, noise, and collinearities among the variables make the estimation of (3.6) complicated and the standard least squares approach often fails totally since the input covariance becomes singular and, therefore, noninvertible. Statistical multivariate regression methods like Partial Least Squares (PLS) regression have been developed to overcome these difficulties and to estimate robust regression models, even from scarce sets of data.

The PLS regression model can be computed using different algorithms, for example, NIPALS (Nonlinear Iterative Partial Least Squares) or Kernel algorithms (see, e.g., [8, 79]). For example, in (one version of) the Kernel algorithm, the *unscaled* input covariance and the data cross-covariance matrices, $M_{\Theta\Theta,1}$ and $M_{\Theta Q,1}$, and the so-called *Kernel matrix*, $M_{\text{Kernel},1}$, are computed in the first step such that (assuming that the data is appropriately mean-centered and scaled),

$$\begin{aligned} M_{\Theta\Theta,1} &= \Delta\Theta^T \Delta\Theta \\ M_{\Theta Q,1} &= \Delta\Theta^T \Delta Q \\ M_{\text{Kernel},1} &= \Delta\Theta^T \Delta Q \Delta Q^T \Delta\Theta = M_{\Theta Q,1} M_{\Theta Q,1}^T . \end{aligned} \tag{3.26}$$

Thereafter, the i^{th}, $i = 1 \ldots N$, PLS latent variable basis vector called the *PLS weight vector*, ϕ_i, $\dim\{\phi_i\} = n \times 1$, can be computed as the eigenvector corresponding to the

largest eigenvalue of $M_{\text{Kernel},i}$. The i^{th} *PLS loading vectors*, p_i and c_i, $\dim\{p_i\} = n\times 1$ and $\dim\{c_i\} = m\times 1$, corresponding to variables Θ and Q, respectively, are computed such that,

$$\begin{aligned} p_i^T &= \frac{\phi_i^T M_{\Theta\Theta,i}}{\phi_i^T M_{\Theta\Theta,i}\phi_i}, \text{ and,} \\ c_i^T &= \frac{\phi_i^T M_{\Theta Q,i}}{\phi_i^T M_{\Theta\Theta,i}\phi_i}. \end{aligned} \tag{3.27}$$

Finally, the matrices $M_{\Theta\Theta,i}$, $M_{\Theta Q,i}$ and $M_{\text{Kernel},i}$ are updated according to equations,

$$\begin{aligned} M_{\Theta\Theta,i+1} &= M_{\Theta\Theta,i} - p_i p_i^T \cdot z_i^T z_i \\ M_{\Theta Q,i+1} &= M_{\Theta Q,i} - p_i c_i^T \cdot z_i^T z_i \\ M_{\text{Kernel},i+1} &= M_{\Theta Q,i+1} M_{\Theta Q,i+1}^T, \end{aligned} \tag{3.28}$$

where z_i are the *PLS score values* (related to input variables Θ, $\dim\{z_i\} = k\times 1$),

$$z_i = \Delta\Theta_i \phi_i, \tag{3.29}$$

and the matrix $\Delta\Theta_i$ is computed such that,

$$\begin{aligned} &\Delta\Theta_1 = \Delta\Theta \\ &\vdots \\ &\Delta\Theta_i = \Delta\Theta_{i-1}\left(I - \phi_{i-1} p_i^T\right). \end{aligned} \tag{3.30}$$

When enough PLS weight vectors are computed recursively, the PLS regression coefficients are given by the equation,

$$F_{\text{PLS}} = \phi\left(P^T\phi\right)^{-1} C^T, \tag{3.31}$$

where $\phi = (\phi_1 \ \ldots \ \phi_N)$, $P = (p_1 \ \ldots \ p_N)$ and $C = (c_1 \ \ldots \ c_N)$.

Another approach for determining the PLS regression is the *eigenproblem oriented PLS* algorithm presented by Hyötyniemi [34]. He sets the maximization of correlation between input and output oriented latent variables as the goal for the PLS model,

$$\begin{aligned} &\max_{\phi_i,\varphi_i} \frac{1}{k} \cdot \phi_i^T \Delta\Theta^T \cdot \Delta Q \varphi_i, \\ &\text{s.t. } \phi_i^T \phi_i = 1 \text{ and } \varphi_i^T \varphi_i = 1, \end{aligned} \tag{3.32}$$

where ϕ_i and φ_i are i^{th} basis vectors of input and output oriented PLS latent variables. Solving the constrained optimization problem (3.32) results in a pair of eigenproblems,

$$\begin{cases} \frac{1}{k^2}\Theta^T Q Q^T \Theta \cdot \phi_i = \lambda_i \cdot \phi_i \\ \frac{1}{k^2} Q^T \Theta \Theta^T Q \cdot \varphi_i = \lambda_i \cdot \varphi_i, \end{cases} \tag{3.33}$$

where λ_i is the i^{th} PLS eigenvalue. The underlying idea is that ϕ_i and φ_i span the input and output oriented subspaces such that the correlation between Θ and Q is maximally represented by the lower dimensional latent variables. In practice, the PLS regression model is constructed by combining the projection of Θ onto input oriented latent basis with the following MLR regression model from the latent variables onto output space into a single linear matrix operation. It turns out that the input and output oriented eigenvectors, ϕ_i and φ_i, coincide with the *left* and *right singular vectors* obtained from the *singular value decomposition* (SVD) of the data cross-covariance matrix,

$$R_{\Theta Q} = \frac{1}{k}\Delta\Theta^T \Delta Q = \Phi_\Theta \Sigma \Phi_Q^T, \tag{3.34}$$

where $\Phi_\Theta = (\phi_1 \ \ldots \ \phi_n)$ and $\Phi_Q = (\varphi_1 \ \ldots \ \varphi_m)$ are the complete sets of left and right singular vectors, and Σ is a $n \times m$ diagonal matrix containing the singular values σ_i. The significance of a singular vector, ϕ_i, is revealed by the magnitude of σ_i. There exists only $\min(n,m)$ nonzero singular values (at the maximum). The singular values are related to the PLS eigenvalues such that $\sigma_i = \sqrt{\lambda_i}$.

The dimension reduction can be performed with the PLS regression model by incorporating only the $N < \min(n,m)$ most significant eigenvectors in the latent variable basis, $\phi = (\phi_1 \ \ldots \ \phi_N)$. The final PLS model then becomes (see [34] for details),

$$\begin{aligned} \Delta\hat{q} &= F_{\text{PLS}}^T \Delta\theta, \text{ where} \\ F_{\text{PLS}} &= \phi\left(\phi^T \Delta\Theta^T \Delta\Theta \phi\right)^{-1} \phi^T \Delta\Theta^T \Delta Q, \end{aligned} \tag{3.35}$$

or when computed from whitened data,

$$\begin{aligned} F_{\text{PLS}} &= W_\Theta^{-1} F_W W_Q \\ &= W_\Theta^{-1} \cdot \frac{1}{k} \phi_W \phi_W^T \Delta\Theta_W^T \Delta Q_W \cdot W_Q. \end{aligned} \tag{3.36}$$

In this thesis, the eigenproblem oriented PLS algorithm is mainly used. Different PLS algorithms (eigenproblem oriented PLS, Kernel PLS and NIPALS) are compared in one of the case studies presented in Section 5.4.

PLS regression works especially well when it can be assumed that the number of underlying *latent variables* carrying the majority of the relevant input-output information in the system is fewer than the original input and output data dimensions imply. In such a situation, collinearities among input or output variables are accepted or, in fact, expected since $N \leq \min(n,m)$. In practice, N is often interpreted more abstractly as a measure of *dimension reduction* and its optimal value is inferred based on data. Optimization of N searches for a compromise between overfitting (i.e., N is too large causing poor generalization) and underfitting (i.e., too radical dimension reduction).

In the targets of IRT applications there usually exists a lower dimensional latent variable space that can be exploited in the modeling. For example, if a system is modeled more and more accurately, usually its model also involves an increasing number of numerical parameters. At the same time, however, the parameters start becoming collinear. This can be seen, for example, in the numerical approximations of partial differential equation systems if the density of the mesh in *finite element* (or *volume* or *difference*) *methods* is increased. In such systems, it can be assumed that N typically grows substantially slower than n. The benefits from MVR methods compared to MLR were experienced in practice within the case studies that are presented in Chapter 5. Some results from a comparison between different PLS algorithms and MLR are given in Section 5.4.2.

If k is small ($k \approx n$ or even $k < n$) and the outputs are corrupted by noise, the estimation of the MLR model is prevented. If an abundant data set is available it, however, represents the optimal linear model. Between these two extremes, the (suboptimal) PLS model often defeats MLR since the dimension reduction decreases the amount of estimated model parameters and the model, therefore, becomes more robust against noisy data.

There exist several methods for finding a practical value for N, but giving any general guidelines for the selection process is more difficult. For example, visual examination of the eigenvalues (see [34]) can be applied if the human intervention during the modeling process is possible. Within IRT this would be slightly problematic since MVR models are estimated repeatedly during the optimization procedure, which is supposed to run as autonomously as possible.

An automatic selection of N can be based on *cross-validation* (CV) that is a widely used method for model validation and model structure optimization. In κ-fold CV, data sets are divided into κ subsets and one subset is omitted from the modeling process at a time and used for estimation of the chosen error criterion for the *prediction* or *generalization error* of the candidate model. In the case of extremely scarce data, *leave-one-out CV* can be used. It means that modeling is repeated k times and the error criterion is computed based on one sample. It has been discovered that leave-one-out CV is badly suited for validation if k is small [79] (i.e., $k \approx N$, in the authors own experience) which creates a problem – in order to determine N reliably one should know N so that a large enough k value could be used. However, this problem can be avoided if $k > \min(n,m)$ samples can be applied. Then leave-one-out CV gives a practical and easy-to-implement solution for optimization of N.

Jackknifing is another method that can be applied for evaluation and comparison of candidate models. Wold *et al.* propose it (along with CV) for model validation. It can be used for estimating the *standard errors* and *confidence intervals* for the model parameters, F_{ij}, of the estimated PLS model [79]. The theory concerning the exact calculation of the confidence intervals for PLS model coefficients is still immature and beyond the scope this thesis.

Recursive modeling using exponential smoothing

If a smooth and continuous J can be assumed, the underlying dependencies between θ and q in certain $\Re_S(K)$ are likely to resemble those observed in $\Re_S(K-1)$, at least to some extent. Therefore, the number of time consuming function evaluations during the optimization procedure can be decreased by utilizing recursive modeling techniques.

In [8] different formulations for PLS are given including a recursive Kernel PLS algorithm that utilizes exponential smoothing to update recursively the data covariance matrices. The eigenproblem oriented PLS can be computed with the same principle in which case the SVD in equation (3.34) is computed upon the recursively updated data cross-covariance matrix,

$$R_{\Theta Q}(K+1) = \mu R_{\Theta Q}(K) + (1-\mu)\Delta\Theta^T \Delta Q, \tag{3.37}$$

where μ is the forgetting factor, $0 < \mu < 1$. The practical value for μ (e.g., $0.7 < \mu < 0.9$) depends largely on the sample size k. Since the gradual updating of the nominal point, $\bar{\theta}$, is not taken into account in (3.37), large values of μ, $\mu > 0.95$, should be avoided unless the number of fresh data samples is significantly low. After updating (3.37), F_{PLS} can be computed similarly according to (3.34) and (3.35). Recursive modeling reduces the number of required function evaluations during the optimization without deteriorating its statistical reliability, since each local estimate can be computed from a smaller data set. It can be seen as an alternative to the moment method – now, it is not only the gradient vector, but the whole cross-covariance matrix instead that is inherited from the previous iterations.

Equation (3.37) still retains some features of the original IRT algorithm since the covariance structure is updated in a batchwise manner. Therefore, it will be called *semi-recursive IRT* in the following. The completely recursive formulation of the IRT algorithm starts to resemble somewhat the sequential stochastic search methods presented in Section 2.3. In the following, this version of the algorithm will be referred to as the RIRT method. In this algorithm, the updating of covariance structures is enhanced by taking into account the update of the nominal point such that,

$$R_{\Theta Q}(K+1) = \mu\left(R_{\Theta Q}(K) + \Delta\bar{\theta}\Delta\bar{q}^T\right) + (1-\mu)\left(\Delta\theta(K+1)\Delta q(K+1)^T\right) \tag{3.38}$$

where the new data point is centered with respect to the new nominal point, $\bar{\theta}(K+1)$ and $\bar{q}(K+1)$, and the old cross-covariance matrix is corrected with the term,

$$\Delta\bar{\theta}\Delta\bar{q}^T = \left(\bar{\theta}(K+1) - \bar{\theta}(K)\right)\left(\bar{q}(K+1) - \bar{q}(K)\right)^T. \tag{3.39}$$

The new nominal point is computed after each sample,

$$\begin{aligned} \bar{\theta}(K+1) &= \mu\bar{\theta}(K) + (1-\mu)\theta(K+1) \\ \bar{q}(K+1) &= \mu\bar{q}(K) + (1-\mu)q(K+1), \end{aligned} \tag{3.40}$$

where $\theta(K+1)$ is sampled from the half-space pointed by the negative cost function gradient. This guarantees that the nominal point is gradually shifted towards the (local) optimum. In RIRT, the update step length is, therefore, inseparably connected to the forgetting factor (reducing the number of method parameters that need to be specified somehow). The input and output data covariance matrices, R_Θ and R_Q, are needed in estimation of the local linear model and in the scaling of data, and they are updated similarly as the cross-covariance matrix in equation (3.38).

The *effective sample size*, k_{eff}, helps to determine an appropriate value for μ,

$$k_{\text{eff}} = \frac{1}{1-\mu}, \tag{3.41}$$

giving, for example, $k_{\text{eff}} = 100$ for $\mu = 0.99$, and $k_{\text{eff}} = 5$ for $\mu = 0.8$. If the equation (3.37) is applied, the sample size, k, influences the effective sample size such that,

$$k_{\text{eff}} = k \cdot \frac{1}{1-\mu}. \tag{3.42}$$

3.2.4 Update principle

In the following, the use of the local linear model in calculation of the update step is presented. First, the managing of multiple targets of optimization is considered, and thereafter, some guidelines for selecting appropriate length for the update step are given.

Decision making

Managing multiple targets in multiobjective optimization is called *decision making*. Targets may be mutually conflicting, and a satisfactory compromise is searched for in such cases. Often the compromise that is found is not the unique optimum in the mathematical sense, but rather belongs to the set of *Pareto optimal solutions* that are equally desirable. The solution $\theta^* \in \Re$ is said to be Pareto optimal if there does not exist another $\theta \in \Re$ such that $q_i(\theta) \leq q_i(\theta^*)\ \forall\ i = 1, \ldots, m$ and $q_j(\theta) < q_j(\theta^*)$ for at least some values of j [58].

In order to find a unique solution among the Pareto optimal solutions, human decision making is needed. Decision making strategies can be divided roughly into three categories [59]: *a priori*, *a posteriori* and *interactive methods*. In a priori methods, the target functions are made somehow commensurable, and using a scalarization function g, the problem is transformed into a single target optimization problem. In a posteriori methods, the decision making process is used *after* the Pareto optimal set (or part of it) has been determined. Thirdly, in interactive methods the decision making is performed concurrently with the optimization. The classification is not strict and, for example, the scalarization applied in equation (3.7) can be considered as either an a priori or an interactive approach, depending on whether the weights are constant or modified during the optimization. It has also the advantage that it enables interactive decision making without necessitating it.

In control parameter tuning, the decision making is, more naturally, an interactive process since finding in advance such weight values that lead to the most desirable solution is practically impossible. In model parameter tuning the commensurability is somewhat easier to determine beforehand. One way is to use the inverse variances of the measured reference signals as weights. In this way, the decision maker places bigger accuracy expectations to those simulated signals that can be measured reliably also from the existing process and, vice versa, quality measures having noisy and unreliable reference signals are not weighted heavily in the scalarization. For example,

$$J = w^T q = \sum_{i=1}^{m} w_i q_i = \sum_{i=1}^{m} \frac{\frac{1}{T}(r_i - y_i)^T (r_i - y_i)}{\sigma_{r_i}^2}, \tag{3.43}$$

presents a scalarization of m MSE quality measures (see Section 2.1.2), where the reference, r_i, and the model output, y_i, are $T\times 1$ (time series signal) vectors.

If the reliability of measurements is unknown, another possibility is to determine *commensurable target units,* s_i, for each model output, that standardize the accuracy targets of signals in different units. A stopping criterion for the optimization can be based also on the commensurable target units. For example, determining that an error of $s_i = 2$ bar in the variable y_i is considered as good estimation accuracy, as an error of $s_j = 1$°C in the variable y_j, in which case scalarization can be accomplished as,

$$J = \sum_{i=1}^{m} w_i q_i = \sum_{i=1}^{m} \frac{\frac{1}{T}(r_i - y_i)^T (r_i - y_i)}{s_i^2}. \tag{3.44}$$

In addition to the initial weighting of targets, the weights can be adjusted also during optimization, for example, based on their distance from the (theoretical) optimum. The role of interactive decision making becomes more important in the vicinity of the optimum (or the set of Pareto optimal solutions) when the optimization starts to slow down due to the contradiction of quality measures. Changing the weight values changes the location of the optimum in the decision space of the scalarized single target optimization problem. The contradictory quality measures can be analyzed, for example, by computing the following *cosine matrix*,

$$\begin{aligned} F'^T F' &= \begin{pmatrix} f_1^T f_1 & f_1^T f_2 & \cdots & f_1^T f_m \\ f_2^T f_1 & f_2^T f_2 & & \\ \vdots & & \ddots & \\ f_m^T f_1 & & & f_m^T f_m \end{pmatrix} \\ &= \begin{pmatrix} \cos(f_1, f_1) & \cos(f_1, f_2) & \cdots & \cos(f_1, f_m) \\ \cos(f_2, f_1) & \cos(f_2, f_2) & & \\ \vdots & & \ddots & \\ \cos(f_m, f_1) & & & \cos(f_m, f_m) \end{pmatrix}, \end{aligned} \tag{3.45}$$

where f_i is the i^{th} column of F' which is the local linear mapping matrix whose column vectors are scaled to unit length. Elements close to one in (3.45) represent parallel targets, zeros indicate orthogonality and negative values reveal conflicting quality measures.

Length of the update step

In practice, the length of the computed gradient vector $||\nabla J(K)||$ needs to be taken into account when the $d(K)$ is computed in equation (3.9). In general, $||\nabla J(K)||$ is zero in the extremum points of $J(\theta)$ and it grows as the slope of $J(\theta)$ gets steeper. Since it cannot be assumed that the $||\nabla J(K)||$ decreases monotonically towards the optimum and that calculations are based on its data-based approximation, it is practical to determine the step length coefficient, γ, in equation (3.9) such that,

$$\begin{aligned}\gamma &= \alpha \cdot \frac{\bar{D}}{D(\nabla J, \mathbf{0})} \\ &= \alpha \cdot \frac{\bar{D}}{\sqrt{\nabla J^T R_\Theta^{-1} \nabla J}},\end{aligned} \tag{3.46}$$

where α is an arbitrary scalar coefficient, $D(\nabla J, \mathbf{0})$ is the Mahalanobis length of the update step (see equation (3.12)) and $\bar{D}$ is the average Mahalanobis distance of Θ from the current nominal point $\bar{\theta}$,

$$\begin{aligned}\bar{D} &= \frac{1}{k}\sum_{i=1}^{k}\left(D\left(\theta(i), \bar{\theta}(i)\right)\right) \\ &= \frac{1}{k}\sum_{i=1}^{k}\left(\sqrt{\Delta\theta^T(i) R_\Theta^{-1} \Delta\theta(i)}\right) \\ &= \frac{1}{k}\sum_{i=1}^{k}\left(\sqrt{\Delta\theta_w^T(i) \Delta\theta_w(i)}\right) \\ &= \frac{1}{k}\sum_{i=1}^{k}\left\|\Delta\theta_w\right\|.\end{aligned} \tag{3.47}$$

The Mahalanobis distance is a distance measure that weights the vector lengths in different directions according to the covariance structure of the distribution. Using (3.46) prevents the new nominal point, $\bar{\theta}(K+1)$, from escaping beyond $\Re_S$.

Another possibility is to connect γ with the limits of $\Re_S$, ρ, such that,

$$\begin{aligned}\gamma &= \frac{\alpha}{\left\|\nabla J(K)\right\|} \cdot \mathrm{P} \\ &= \frac{\alpha}{\left\|\nabla J(K)\right\|} \cdot \begin{pmatrix} \rho_1 & & 0 \\ & \ddots & \\ 0 & & \rho_n \end{pmatrix},\end{aligned} \tag{3.48}$$

which scales $d(K)$ to reach exactly the surface of a spherical (or elliptical) $\Re_S$ when $\alpha = 1$. Notice that now γ is a $n \times n$ diagonal matrix. The drawback of both presented methods is that the successive domains become partially overlapping (with moderate values of α, i.e., $\alpha < 2$) which makes the optimization procedure unnecessarily slow. To speed up the tuning procedure, $\Re_S$ can be widened by increasing ρ or α.

In the classical nonlinear LO methods, for example, in Newton's method, optimal step length is usually computed using the Hessian of the cost function, or it is determined based on a rigorous *line search* [61]. Reliable utilization of the Hessian information requires in practice that analytical gradient information is available and that the hyperparabola shape for J can be assumed. Therefore, this approach is questionable in this context. However, since in the neighborhood of an optimum any nonlinear continuous function can be approximated by a hyperparabola [61], the Hessian approach could be applied to improve the converge of the optimization on its final iteration steps. Line search methods are not applied here since extrapolation far beyond $\Re_S$ is neither

safe nor practical. The linear cost function estimate is valid only locally and the observations of q are not deterministic which complicates, or prevents, the use of line search, especially when IRT is run in adaptive mode (see Chapter 3.2.3). From the practical point of view, line search is inconvenient to apply with simulation models that cannot tolerate large abrupt parameter changes. Furthermore, recursive modeling, and especially the completely recursive formulation of IRT, make this problem less crucial since the nominal point is gradually shifted towards the optimum.

3.2.5 Summary of IRT and RIRT algorithms

Applying IRT or RIRT for control or model parameter tuning begins with the formulation of tuning targets into the form of m quality measures, q, and giving the appropriate weightings, w_i, for q_i. Secondly, the decision variables θ, their initial values and the absolute maximum and minimum values are determined. Thirdly, an appropriate sampling method is defined including local limits of variation and other distribution parameters. For the batch-wise formulation of IRT also the sample size k is specified. Finally, the maximum number of optimization steps is given after which the tuning can be started. In the following, the summaries of running IRT and RIRT algorithms are given.

Initialization and iteration procedure of the IRT (and semi-recursive IRT) algorithm(s)

```
1. Define q_i and w_i
2. Specify θ, and determine θ̄(K=0) and ℜ
3. Determine K_max
4. Select sampling distribution for θ
          Define initial distribution parameters
          Determine ℜ_s and k
5. Start iteration, K = 0:

while (K ≤ K_max and J(K) > J_target) {
          Sampling:
                    Create k decision variable combinations, Θ
          Function evaluations:
                    Simulate system performance using each Θ(i,:)
                    Compute performance evaluation, Q(i,:)
          Modelling:
                    Preprocess Θ and Q
                    Compute F, select N using cross validation
                    Gaussianity testing of residuals
                    if (Residuals are non-Gaussian)
                              Decrease ℜ_s
          Update step:
                    Compute θ̄(K+1)
                    if (Semi-recursive IRT is used)
                              Update R_ΘQ
                              Update distribution parameters
                    K = K + 1
}
```

Initialization and iteration procedure of the RIRT algorithm

```
1. Define q_i and w_i
2. Specify θ, and determine θ̄(K=0) and ℜ
3. Determine k_max and γ
4. Select sampling distribution for θ
          Define initial distribution parameters
          Determine ℜ_s
5. Start iteration, K = 0:

while (K ≤ K_max and J(K) > J_target) {
          Sampling:
                    Pick θ(K+1) from the sampling distribution
          Function evaluation:
                    Simulate system performance using θ(K+1)
                    Compute evaluation based on results, q(K+1)
          Modelling and update step:
                    Preprocess θ(K+1) and q(K+1)
                    Update R_θQ(K+1), R_θ(K+1) and R_Q(K+1)
                    Update θ̄(K+1) and q̄(K+1)
                    Compute F, select N using cross validation
                    Gaussianity testing of residuals
                    if (Residuals are non-Gaussian)
                              Decrease ℜ_s
                    K = K + 1
}
```

3.3 Constraints vs. degrees of freedom

The applicability of the IRT method presented in Sections 3.1 and 3.2 to parameter optimization of large-scale technical systems can be justified if the most practical way to model the dependency between system parameters and quality measures is considered [21, 35, 36].

Neocybernetics is a novel framework for studying large interconnected multivariate systems of biology, ecology and economy proposed by Hyötyniemi [36]. Even though advanced technical systems, like industrial processes under modern process control or large-scale dynamic process models, are more like constructivistic than neocybernetic systems, they still possess enough internal feedback couplings and other cohesive constraints so that they follow certain principles of neocybernetics. The essence of the system is captured in the emergent quality measures, and studying the degrees of freedom in the system presents the most practical way for improvement of system performance.

In mathematics (in linear algebra) it is said that a system of m equations involving n variables possesses $n - m$ *degrees of freedom*. The equations determine the internal structure of the system, i.e., after some variables have been fixed they start to *constrain* the values of the remaining variables. The remaining degrees of freedom appear as variation potential in the values of the constrained variables.

Respectively, the traditional view of modeling dynamic industrial processes means estimation of the constraint equations among the system variables v,

$$v(k) = \left(y^T(k) \quad y^T(k-1) \quad \cdots \quad u^T(k) \quad \cdots \quad r^T(k) \quad \cdots \right)^T . \tag{3.49}$$

All dynamic model structures that are linear in parameters can be presented as a set of linear equations,

$$\Gamma^T v = \mathbf{0} . \tag{3.50}$$

For example, an elementary control algorithm can be presented as,

$$u_i(k) = \alpha \cdot \left(r_i(k-1) - y_i(k-1) \right), \tag{3.51}$$

and a large set of dynamical processes can be approximated with time series models,

$$y_j(k) = \sum_{\iota} \alpha_\iota y(k-\iota) + \sum_{\kappa} \beta_\kappa u(k-\kappa) \;\; \forall \iota, \kappa \in \{ \mathbb{N} : \iota \neq 0 \} . \tag{3.52}$$

Each control algorithm increases the amount of constraints among the system variables and, at the same time, decreases the number of degrees of freedom by reducing the uncontrolled variation which is manifested by the quality measures. In a way, a modern industrial plant with developed control structures can be seen as an outcome of *artificial evolution* that finally starts to resemble a neocybernetic system (see [36] for a more detailed description). In such systems, uncontrolled variation remains only in a few directions of the parameter space. This can be seen as a motivation for concentrating only on the modeling of degrees of freedom instead of the constraints determining the system structure and keeping the system stable.

The quality measure variation that is obtained by varying system parameters reveals the potential for performance improvement via parameter tuning, whereas the detailed constraint based system model determines only the nondescript mean value of the variation. Achieving the maximal variation of quality measures is not the target. Instead, one searches for the direction of maximum covariation between the system parameters and the quality measures. If the system is interconnected and constrained enough, low dimensional models concentrating on the degrees of freedom can be obtained which are capable of describing these relevant underlying dependencies. This information can be applied to push parameters into directions that improve the system performance. This type of parameter optimization can be interpreted as a higher level optimizing controller of the overall system.

3.4 Comparison to other controller tuning and parameter optimization methods

Most of the traditional controller tuning techniques presented in Section 2.2.2 can be applied only in a reductionist setting concentrating on one controller at a time. Also, the numerical controller tuning techniques in Section 2.2.3 take insufficiently into account the high dimensionality of the problem. When controller tuning is seen as general data-based optimization problem, several possible methods are available. Now the

characteristics of the optimization problem suggest which method can be successfully applied. Quantitative comparisons of efficiency of different optimization methods are difficult to perform in practice. Comparisons should be repeated several times with different problems in order to get reliable results.

First of all, use of the more sophisticated nonlinear LO methods presented in Section 2.3 is prevented in this context by the fact that gradients are data based approximations. Reliable use of Hessian information requires, in practice, that at least the gradient information should be available in analytical form [61]. The advanced nonlinear LO methods also necessitate accurate line search which in this case is problematic.

Secondly, exact GO methods are difficult to apply here since the parameter space sampling is in practice restricted to a local domain (see Section 3.2.2). In control parameter tuning, it is not acceptable that the system is disturbed by large abrupt parameter changes because of the risk of running the process into instability. Also the plant wide dynamic simulation models are often sensitive to large parameter variations and it is, therefore, more practical to use only local perturbation to prevent divergence of the solver algorithm.

Thirdly, heuristic GO methods like SA resemble the IRT method in many ways. Their basic formulations, however, do not typically consider the problems related to the high dimensional data or dimension reduction in any way. In the IRT method, MVR methods are applied to overcome these problems and the filtering of noise as well. The completely recursive formulation of IRT approaches the sequential stochastic search methods incorporating also the advantages of using MVR methods. In addition to the mere parameter update direction, the local linear model $\Delta q = F^T \Delta \theta$ makes it possible to analyze the underlying interdependencies of multiple targets, i.e., their parallelism, independence, and so on.

4 IMPLEMENTATION AND APPLICATION OF IRT

In the following, description of the IRT method's implementation in a simulation assisted automation testing environment is first given in Section 4.1 along with a clarification of the research focus of the related research projects. Secondly, the application of IRT in different contexts is discussed.

4.1 Implementation of IRT method

Application of the IRT method involves many practical arrangements concerning the configuration and execution of simulation sequences, the storing and viewing of simulation results, connecting separate models into a synchronous simulation cluster, etc. Therefore, using IRT becomes much more comfortable and less error prone if the algorithm is implemented into a suitable platform designed for simulation assisted engineering. The research projects Testing Manager and Simbiot, which have set up the framework for this thesis, in many respects have both concentrated on this topic. In the following, the emphasis is mainly on the IRT algorithm and the Tuning Tool developed during these projects, even though other profound research interests were also present within the projects.

In the Testing Manager project the research focused on simulation assisted automation testing including, for example, identification of new simulation assisted working practices and their benefits in different phases of the automation life cycle, the study of requirements for synchronous communication of different software components and managing the simulation sequences of simulation clusters, consisting of several separate model components. During the project, control parameter tuning using the IRT method was tested on large scale industrial simulators and the algorithm was developed further. A requirement specification for the Tuning Tool (see below) operating in the Testing Station environment was made. The Testing Station is a simulation assisted automation testing platform that combines dynamic process simulation models and virtual automation systems into a synchronous simulation cluster and provides the user with other supplementary tools like connection to historian data base and visualization tools for viewing simulation result, among others [43, 75]. The implementation of Testing Station is based on the results of many earlier and concurrent research projects of VTT with several research and industrial partners, for instance, within the PI, ÄLY and MASI technology programs of TEKES.

The Simbiot project (as a part of the Semserv project consortium) focused on studying the prerequisites, methods and advantages of *plant model oriented* (PMO) *process design* in which all design and maintenance information is incorporated into a plant model during the design, implementation and operation phases of the process plant life cycle. During the project model parameter tuning, use of the IRT method was tested and a tool for model and control parameter optimization called the Tuning Tool including the IRT method was implemented and connected to the Testing Station.

4.1.1 Simulation assisted process and control engineering

Information management during the entire process plant life cycle has become an active research topic due to the increasing demands for flexible data exchange and intelligent design and maintenance information processing. PMO engineering is an approach promoting this goal. Combining the design information into a plant model improves the communication between the design engineers of different fields, reduces the amount of manual work and the number of errors during the design process and opens up new model-based working practices for process and control design and maintenance. Several attempts have been made to find general standards for the presentation of plant model information but none of them has yet reached a dominating position [43]. The plant model can be defined in several ways, but in principle it is a model that describes the structure, operation and relations of the process and automation components of a plant. Technically speaking, the model consists of different process objects having certain properties and relations between these objects. Details concerning the plant model definition and implementation are beyond the scope of this thesis but the new simulation assisted working practices for process and control engineering increase the demand for parameter tuning methods like IRT.

In PMO engineering, the simulation assisted working practices become an inseparable part of the overall design process. Traditionally, process simulation has been a separate task along (or after) the actual design process carried out on specific modeling software without concrete connections to the design environment. However, many design phases could benefit considerably from simulation and computational design principles. Several potential uses of simulation during an automation delivery project have been reported in [44, 49]. For example, the rationality of system requirements can be verified, different process and control designs compared and later on the combination of chosen solutions evaluated before implementation using simulation. Also, the operation and maintenance phases benefit from the overall system model since simulations can substitute for expensive and possibly hazardous process experiments, different operation strategies or process changes can be tested with the simulator before introduction into the existing system.

To reach the above mentioned potential benefits, new design tools are required and conventional working practices need to be revised. First of all, the design environment, i.e., the plant modeling software, should support effortless model parameter estimation. In order to compare objectively different process structures, control solutions or operation schemes, their performance should be first optimized according to the chosen performance criterion. As the complexity of the compared systems and the number of candidates increases, an efficient parameter tuning tool becomes indispensable.

Secondly, the plant model produced during the design phase is still only a rough generalization of the existing system. Even though this preliminary model is useful in

many dynamic studies preceding the system implementation, it is highly beneficial to improve the estimation accuracy of the plant model using parameter estimation as soon as measurement signals from the existing system are available.

Thirdly, the plant model requires maintenance during the plant life cycle in the same way as the plant itself. If updating the model is neglected it quickly becomes obsolete and the modeling efforts are lost. Slow changes like fouling or wear of process equipment and, for example, addition of completely new process equipment or control structures, need to be taken into account in the plant model.

All three above presented aspects advocate the inclusion of a parameter tuning tool (along with other assisting design instruments) in the PMO design environment. On the one hand, the idea of PMO design looses most of its attraction if the model can be applied only as storage for design information (due to inadequate estimation accuracy). On the other hand, the need to connect the plant model to external optimization software for parameter tuning easily ruins the usability of the plant model environment (and probably also its popularity among the design engineers), which corresponds to the current situation on the field quite well. Successful PMO design practice would end the discussion about the costs of dynamical process modeling as the plant model would be an inseparable part of the process plant, similar to the automation and information management systems found today.

4.1.2 Tuning Tool

During the Simbiot and Kelo-VTT research projects the Tuning Tool (including the IRT algorithm) was implemented in the Testing Station environment. The underlying application service framework architecture is specified in detail in [48] and its use for simulation assisted automation testing is presented in [75]. The Testing Station enables the configuration and controlling (running, stopping, loading, saving, etc.) of test sequences that consist of different process events, operator interactions and data collection phases with the chosen simulation model (or simulation cluster consisting of several separate interconnected models). Models are connected to the Testing Station with the OPC XML Data Access (DA) and Data Exchange (DX) specifications. The Testing Station environment also facilitates visualization and archiving of the simulation results.

The Matlab code of the IRT algorithm was turned into a C shared library file using the Matlab Compiler product. A simple driver code in C was created for calling these library functions and returning the computed results. Since the Tuning Tool and Testing Station are implemented using Java, the necessary Java Native Interface (JNI) classes and the C wrapper code (enabling the communication between C and Java) were created using the SWIG (Simplified Wrapper and Interface Generator). Finally, the whole thing was compiled into a dynamically linked library file (dll) using the *mbuild* command in Matlab. The resulting library file can be called directly from the Java code of the Tuning Tool.

Figure 2 presents the starting view of the Testing Station. The upper part is the tuning control bar of the Tuning tool. It is used for managing (i.e., starting, stopping, etc.) the tuning procedure and launching visualizations (see Figure 3). The tree structured workspace on the left contains most of the functionalities needed for the initialization of the tuning case. For example, details concerning decision variables, quality measures and

parameterization of the IRT method can be modified by editing the properties pages of the tree nodes.

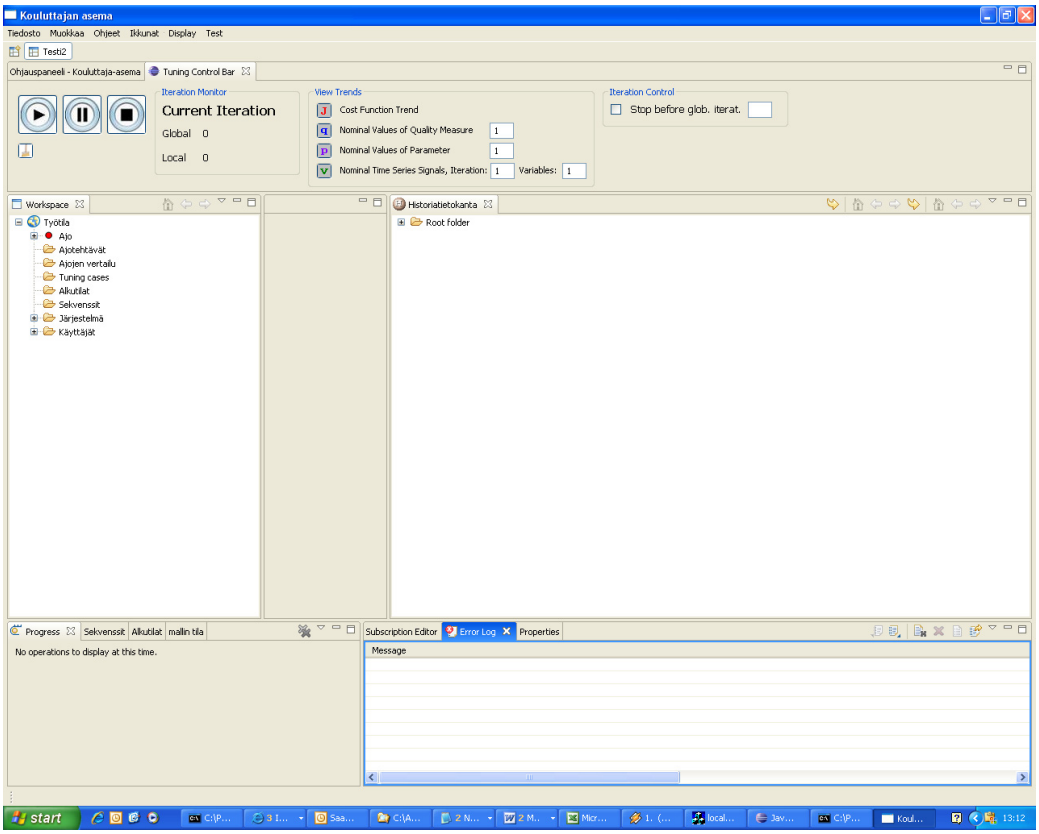

Figure 2. Testing Station and the tuning control bar of Tuning Tool.

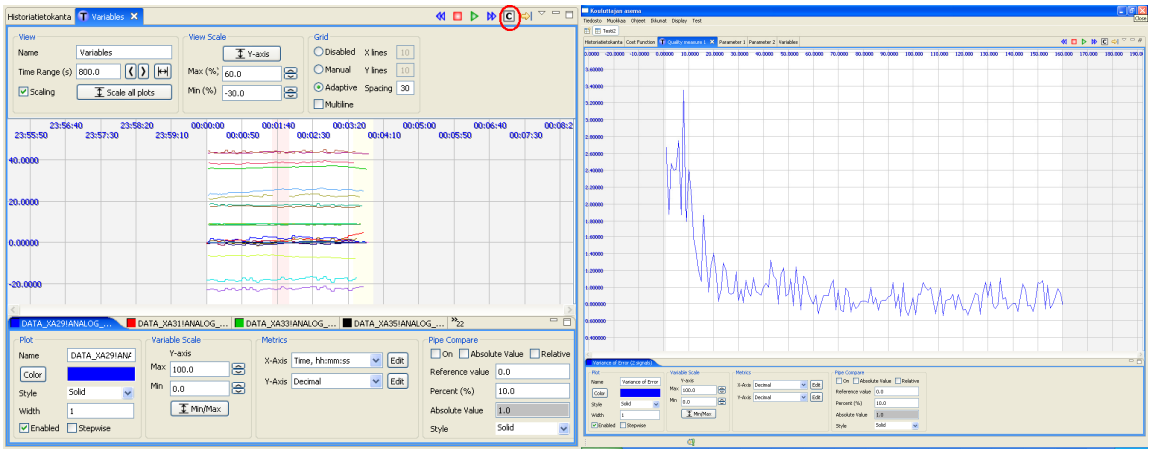

Figure 3. Visualizations of simulated time series signals (left) and quality measure values (right).

The Testing Station involves also views of the components of the connected simulation models (see Figure 4) and contents of the historian data bases using OPC. The Error log of the Testing Station shows reports on the status of the tuning procedure (lower right corner in Figure 2) and the sequence editor makes it possible to create suitable simulation sequences for the tuning (Figure 5).

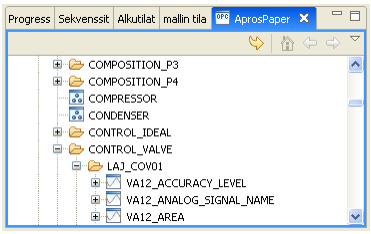

Figure 4. View on the components and component properties of the connected Apros Paper model.

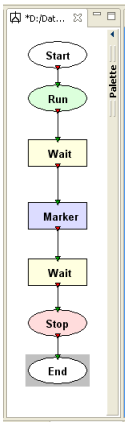

Figure 5. A simulation sequence containing essentially two wait blocks and a marker block denoting the beginning of the data logging period.

The Testing Station is under continuous development work and the above figures are examples showing the appearance of the UI in the spring of 2008. To get the full benefit from the tuning environment, for example, more visualization tools should be implemented to support visual data analysis and error detection during the tuning. For example, histograms of quality measures are a practical way to detect problems resulting in outliers.

4.2 Application of IRT

There are several possible arrangements in which the IRT method can be applied. Targets of tuning can be either control and other numerical operational parameters of the existing system or the parameters of a model. In the following, three different scenarios are presented. Two of them consider the simulation assisted use case and the third one presents a direct application of IRT in an existing system. The application of IRT in different situations has been discussed also in [22].

4.2.1 Model accuracy improvement

Dynamical process modeling may have many different goals. No matter what the purpose of the model is, improving its accuracy makes the following analysis more valuable in any case. The IRT method pulls together suitable tools for tuning model parameters of large dynamical process models. Application to other models, if considered practical, is also possible. In practice, if measurement data from the system is available model

parameters can be optimized by comparing the simulated output estimates to the measured outputs when the model is run with the measured inputs (see Figure 6). The improved accuracy of the plant model opens new possibilities for its usage and, eventually, model based engineering can be seen as an iterative process incorporating alternating phases of model and system parameter tuning steps (see Figure 7).

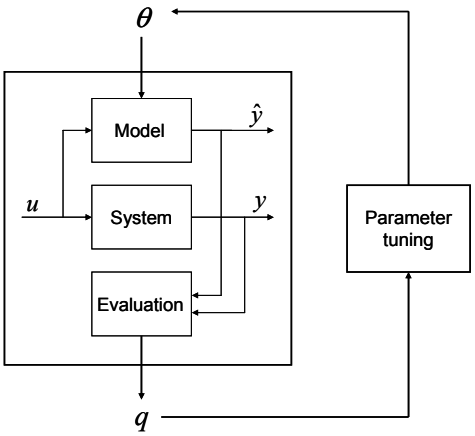

Figure 6. Model parameter tuning using measurent data as reference.

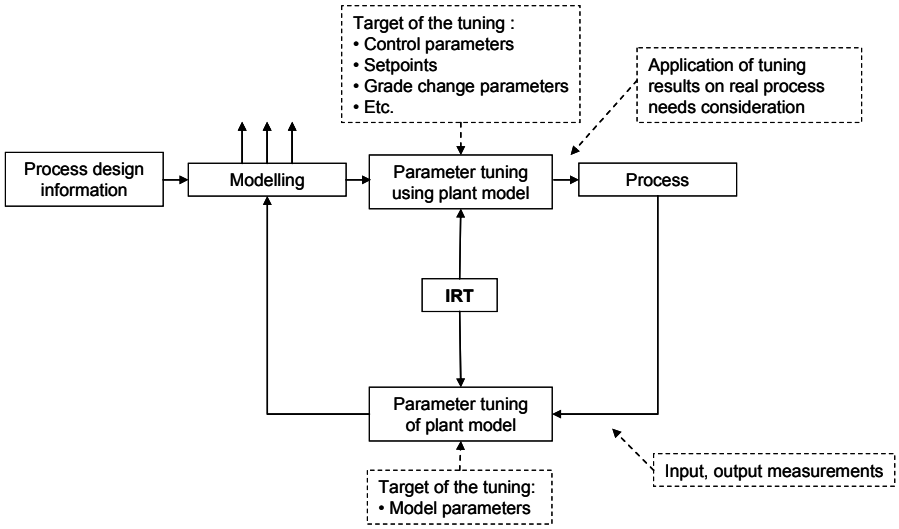

Figure 7. Using IRT to parameter tuning in different phases of model based engineering.

4.2.2 Controller and process design

Finding efficient techniques for designing control systems for large-scale continuous plants with a large number of manipulated and controlled variables is one of the challenges industry is facing today [9]. Therefore, modern controller design concentrates on developing large control concepts for different process entities, for example, headbox or profile control structures for paper machines, large MPC controllers applied on oil refinerys, steam network controls, and so on, because it is not practical nor economical to start the controller design from scratch for every delivery. Dynamic simulation is nowadays an established instrument in controller design and product development since

it is inevitably the easiest way to test, demonstrate and evaluate the performance of large dynamic systems.

In order to compare objectively several alternative control solution candidates, their performance needs to be optimized first. Therefore, an efficient and effortless tuning method is needed that can be applied for tuning the parameters of the competing control structures. Since IRT can be applied without considering the type of the control concepts (MIMO controller, network of SISO controllers, MPC, etc.), it fits well for prototype tuning in a simulation assisted scenario that is presented in Figure 8.

Also, the tuning of existing controllers can be accomplished in the same way if only a model of the controlled process exists. Tuning controllers on a simulator has several advantages, for example, the production of the existing system is not disturbed with experiments and controllers can be tested against different severe disturbances without a risk of damaging the process machinery. Introducing the simulated results into practice needs to be done carefully since the simulation model is naturally only an approximation of the true system. Simulated results, however, give an insight into which direction the parameter values in use should be changed in order to improve the performance.

Respectively, IRT can be applied within simulation assisted process design. Alternative new process solutions can be tuned to their maximal utility from the overall performance point of view before making any comparisons.

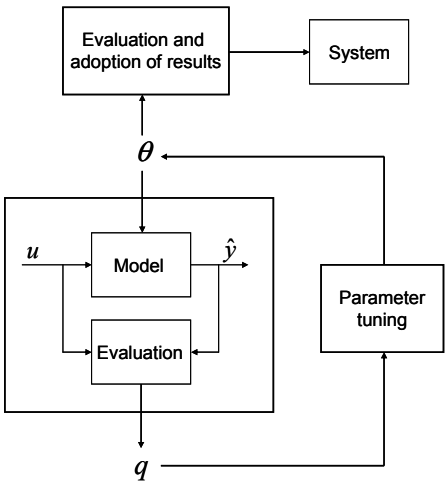

Figure 8. Simulation assisted controlled tuning.

4.2.3 Online optimization of process operation

The IRT method can be applied as an adaptive control structure as well (see Figure 9). If small perturbation of variables is tolerated, IRT can be applied to analyze the statistical dependencies between different parameter value combinations and observed performance. Since the input signals of the system cannot be fully controlled, much longer observation periods and larger data sets are required as compared to the simulation based scenario presented in Section 4.2.2. For the same reason online optimization of repetitive control tasks or changes of operation point, for instance, works out best. Adaptation is not only restricted to the control parameters, as any operational parameters can be used as decision variables.

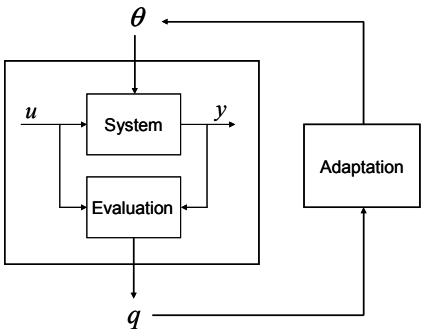

Figure 9. Adaptive application of IRT method.

Another online application possibility relates to the process models applied within MPC. During the normal operation of the MPC controlled system, the model parameters can be tuned iteratively by computing several parallel model responses to the computed control action with different model parameter value combinations. When the actual system response is recorded from the system, it can be compared to the modeled responses and the differences between different parameterizations can be expressed using different quality measure definitions. In this way, IRT can be applied online without disturbing the operation of the true system.

5 CASE STUDIES

This chapter presents several case studies about using the IRT method for control and model parameter tuning. Each section presents briefly the target application, shows the formulation of the tuning case, introduces the obtained results and ends with some discussion. In Section 5.1 the advantages of using the IRT method are demonstrated with an elementary example. Section 5.2 gives an example of using IRT on controller tuning of a continuous pulp digester. The results from a corresponding case study considering controller tuning on a power plant simulation model have been reported in [17, 18]. Sections 5.3 and 5.4 concentrate on the model parameter tuning using IRT.

5.1 Introductory example of interacting SISO controllers

In the following, an example using a simple Matlab model is given to introduce the research problem concerning control parameter tuning in practice. A similar simple Matlab case study about using the IRT method for model parameter tuning is presented in [20]. It must be noted that with simple examples like these two, the most essential advantages of the IRT method cannot be demonstrated and many other optimization algorithms could be used here successfully as well.

Systems with multiple inputs and multiple outputs (MIMO) tend to have more or less severe interactions between the process variables. If the controller design is conducted to individual input output pairs, the resulting system will not reach the best achievable overall performance. This is demonstrated here with a simple example.

Let us study the following system $G(s)$ with two inputs and two outputs.

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} = \begin{bmatrix} \frac{2}{s+1} & \frac{3}{s+2} \\ \frac{1}{s+1} & \frac{1}{s+1} \end{bmatrix} \tag{5.1}$$

$G(s)$ has two input (control) signals, u_1 and u_2, and two output (controlled) signals, y_1 and y_2. Suppose that y_1 is controlled by manipulating u_1, and y_2 is paired with u_2, and the non-diagonal terms of $G(s)$ are assumed to be zeros. The IMC control design method leads to PI type of SISO control laws,

$$G_{C,1}(s) = \frac{Y_1(s)}{U_1(s)} = \frac{1}{2\alpha_1}\left(1+\frac{1}{s}\right) \tag{5.2}$$

and

$$G_{C,2}(s) = \frac{Y_2(s)}{U_2(s)} = \frac{1}{\alpha_2}\left(1+\frac{1}{s}\right). \tag{5.3}$$

Above, α_1 and α_2 correspond to the closed loop time constants of the two controlled systems

$$G_{CL,1}(s) = \frac{G_{C,1}(s)G_{11}(s)}{1+G_{C,1}(s)G_{11}(s)} = \frac{1}{\alpha_1 s+1} \tag{5.4}$$

and

$$G_{CL,2}(s) = \frac{G_{C,2}(s)G_{22}(s)}{1+G_{C,2}(s)G_{22}(s)} = \frac{1}{\alpha_2 s+1}. \tag{5.5}$$

If similar dynamics are desired for $G_{CL,1}$ and $G_{CL,2}$, (i.e., $\alpha_1 = \alpha_2 = \alpha$), both closed loop time constants, α_1 and α_2, can be chosen, for example, such that the cost function

$$J = \sum_i \sum_{t=0}^{T} \left(y_i(t) - r_i(t)\right)^2 \tag{5.6}$$

is minimized. Above, T is the length of the time series data and $r_i(t)$ is the reference signal of $G_{C,i}$. The Sum of Squared Error (which equals to ISE in a continuous time case) is applied to avoid large deviations from the reference signal. Here the reference signal contained a unit step and a subsequent impulse disturbance which means that the objective was to find a good compromise between step response performance and impulse disturbance attenuation. The above formulation of the controller design task turns out to be a convex optimization problem with a unique minimum with respect to α. Figure 10 presents the cost function values with $\alpha = [0\ldots5]$.

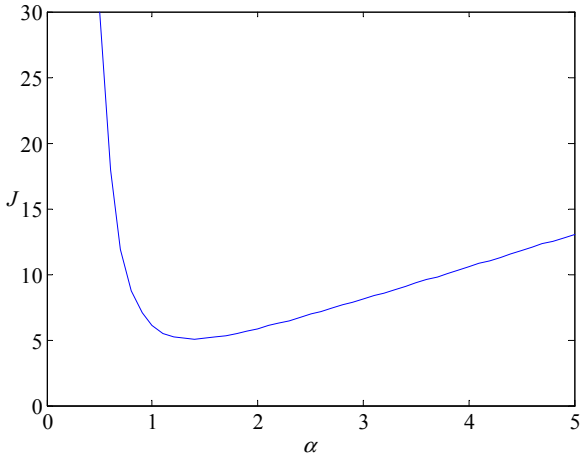

Figure 10. Cost function *J* reaches minimum with $\alpha = 1.4$.

The minimum is reached with $\alpha = 1.4$ which leads to following PI control parameters.

$$\begin{aligned} K_{P,1} &= \frac{1}{2.8} \approx 0.357 \\ T_{I,1} &= 1 \\ K_{P,2} &= \frac{1}{1.4} \approx 0.714 \\ T_{I,2} &= 1 \end{aligned} \tag{5.7}$$

The performance of the system $G(s)$, controlled with two separate controllers, $G_{C,1}(s)$ and $G_{C,2}(s)$, is illustrated in Figure 11. The performance was evaluated with a simulation sequence with two setpoint changes: $r_1(t) = 1$, $t \geq 0$, and $r_2(t) = 1$, $t \geq 60$. Now the interaction terms in $G(s)$ cause interference between the controlled variables. As can be seen, the performance is satisfactory although somewhat sluggish.

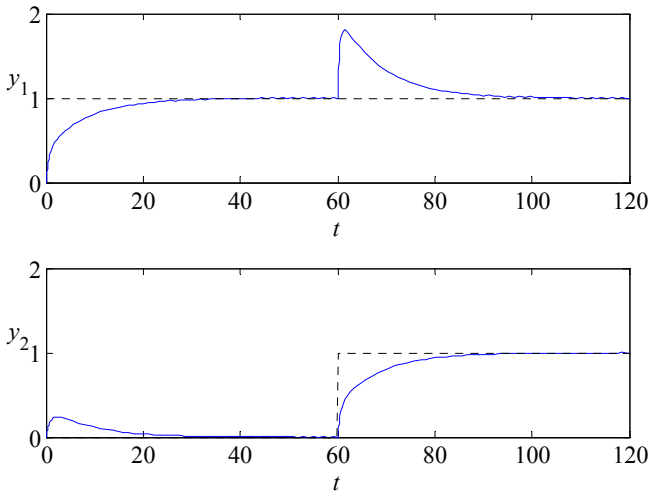

Figure 11. Performance of the system $G(s)$ with two IMC tuned PI controllers, controlled variables (solid blue) and reference signals (dotted black).

The result of the controller design can be improved if the interaction terms in $G(s)$ are taken into account. In a simple case like this, designing a multivariable controller is not difficult. However, as the size of the MIMO system rises, identification of $G(s)$ from the existing system quickly becomes too laborious. Therefore, more advanced controller tuning methods are needed. Further, in practice the process model $G(s)$ is not known exactly but an approximation is used in the controller design resulting inevitably in a non-optimal solution.

The IMC tuning can be improved with the IRT method. Using the same cost function as above but letting the interactions take their effect, a significant enhancement of performance is achieved (Figure 12). Both step responses have become faster and the disturbance in y_1 is attenuated more efficiently. Only a minor compromise has to be accepted in the disturbance attenuation in y_2.

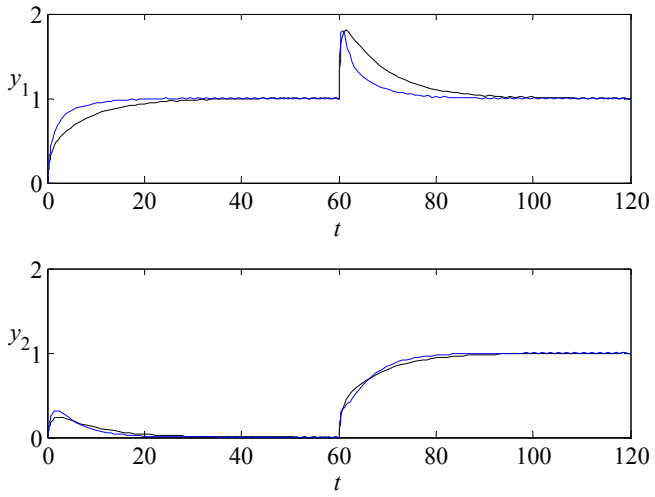

Figure 12. Performance comparison of IMC (black) and IRT tuned (blue) PI controllers.

The improvement was achieved by running a simple version of IRT $K = 30$ global iterations, each consisting k = 20 local iterations. The steepest gradient method was applied with a constant update step size, $\gamma = 1$ (see Section 3.2.4 for other possibilities). Local linear models were calculated with standard MLR. The PI control parameters obtained the following values after the tuning.

$$K_{P,1} \approx 0.5509, \quad T_{I,1} \approx 0.7105$$
$$K_{P,2} \approx 1.1552, \quad T_{I,2} \approx 1.4402$$

The above example illustrates how the overall performance of MIMO systems has an important role in the controller design and controller tuning tasks. In a realistic industrial case, identification of numerous open loop models for $G(s)$ is impossible in practice, and, therefore, an explicit multivariable controller design is not a tempting option in many cases.

5.2 Veitsiluoto pulp digester

Simulation assisted plant wide control performance optimization was tested on a large scale with a simulation model of Veitsiluoto pulp mill [19]. The process model was constructed with the APROS software. Figure 13 gives an overview of the process model. In the following, the necessary terminology related to pulp production is first briefly explained in Section 5.2.1. More details on chemical pulping can be found, for example, from [12]. Then, the pulping process and model of the pulp mill is introduced in Section 5.2.2, after which the tuning targets of the case study are formulated in Section 5.2.3. The results are presented and discussed in Sections 5.2.4 and 5.2.5.

The most important goals of the case study were to test IRT method with a realistic problem, study how the larger data dimensions affect solving the parameter tuning problem and (hopefully) show that also a large complex system can be managed with IRT. The word realistic refers here to the complexity of the process and its model, to the number of the parameters and quality measures to be optimized, and to their

interconnectedness. The continuous pulp digester is a very challenging process from the control engineering point of view – it has a large dead time (of several hours), raw material properties introduce unmeasured disturbances to the system, and the number of chemical reactions taking place in the cooking phase make it difficult to understand all the relevant interactions of the components in the chip-liquor mixture. It is extremely difficult even for a domain area expert to get a clear picture of the process and the underlying causalities because of the large number circulation streams between process components (see Figure 13).

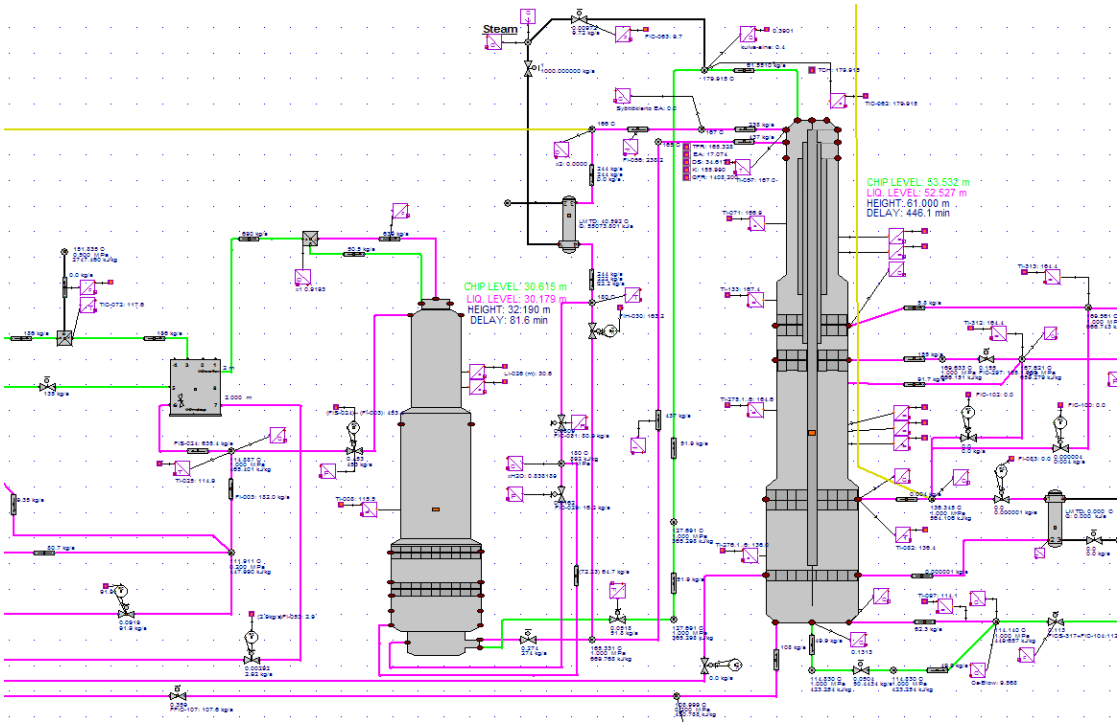

Figure 13. A view of Apros pulp mill model using the Grades user interface: The chip feed screw conveyor (left), the impregnation vessel (in the middle), and the pulp digester (right).

5.2.1 Terminology

The *kappa number* is a measure of lignin content in the cooked pulp. Traditionally the kappa number has been defined based on the potassium permanganate consumption in an acid pulp dilution. In practice, the kappa number is nowadays measured with online kappa analysers which handle the sampling and analysing in a couple of minutes. The lignin content of the pulp determines the brightness of the produced fibres affecting thus the subsequent bleaching stages. Furthermore, it affects the strength properties of pulp, which makes it in practice the most important controlled variable in pulp production.

The kappa number is controlled by adjusting the *H factor*, which is an experimental combination of the cooking time t_T and the temperature T. One H factor unit denotes the effect of one hour of cooking in 100 °C. The H factor is defined as follows.

$$H = \int_0^{t_T} e^{(43{,}2 - 16115/T)} dt \tag{5.8}$$

The equation (5.8) cannot be applied for H factor prediction since the exact temperature profile during the cooking is required. H factor prediction is essential for the kappa number control. Therefore, the H factor can be calculated, for example, by using a simplification

$$H = \sum_{i=1}^{n} H_i = \sum_{i=1}^{n} e^{43.2 - \frac{16115}{T_i}} \cdot \Delta t_i \tag{5.9}$$

in which the digester is divided into n parts that are assumed to have constant temperature. Above, H_i is the i^{th} partial H factor, T_i is the temperature in i^{th} cooking part, and Δt_i is the time the i^{th} cooking phase takes. In absence of temperature measurements, the temperatures T_i can be approximated roughly according to

$$T_i = \alpha_i T_0 , \tag{5.10}$$

in which T_0 is the digester top temperature and α_i is an adjustable parameter.

After the cooking process, the pulp is washed in several successive washing stages before bleaching. The cooking chemicals are recycled, and, therefore, minimization of the washing losses is pursued. They also affect the required chemical dosages in the subsequent bleaching stages. The first washing is performed at the bottom of the digester with a counter current washing. The efficiency of the washing is measured with the *washing coefficient* that is defined as a ratio of used washing liquor to the amount of washed pulp. To ensure uniform pulp properties, a constant washing coefficient is desirable.

During past decades, many theoretical and empirical cooking models for the kraft pulping process have been proposed. One of the simplest models was proposed by Hatton in 1973 (see, e.g., [13]), which predicts the kappa number κ of the produced pulp based on the measurements of the H factor and effective alkali concentration c_{EA},

$$\kappa = \alpha - \beta \cdot \log H \cdot c_{EA}^{\chi} , \tag{5.11}$$

in which α, β and χ are adjustable parameters.

5.2.2 Continuous pulp digester and control problem

A general overview of the model is presented in Figure 13. First, the woodchips and the impregnation liquor are mixed, after which the mixture is fed into the impregnation vessel. From the bottom of the impregnation vessel, the flow continues to the top of the digester, where the mixture is heated with steam to the desired cooking temperature. The wooden substance was modeled to consist of several cellulose, carbohydrate and lignin components, and the liquor was assumed to contain sodium hydroxide and sodium sulphide in addition to the organic compounds dissolved from the woodchips. The chemical reactions during the cooking phase, and mass and energy balances were modeled according to the Purdue model presented in [71]. The chip compression profile during the cooking was calculated as described in [13]. Also several black liquor circulation streams corresponding to the existing process structure of the Veitsiluoto pulp mill were modeled. The washing and bleaching operations subsequent to cooking, however, were excluded from the model.

Here it must be reminded that the simulation model does not correspond to reality in every detail. The most significant simplifications relate to material properties. The model neglects the effects of the chip geometry when calculating the chemical reactions of the cooking phase. In practice, the chip size distributions are important variables that determine the properties of the produced pulp. Therefore, constant (and appropriate) chip size distributions are assumed in this case study, and the largest disturbances to the process are assumed to result from the varying chemical composition of the raw material and from the changes of the heating steam properties. Despite the simplifications, the simulation model describes the most relevant properties and functionalities of the Veitsiluoto pulp mill with an acceptable accuracy making the case study realistic enough to be convincing. What is more, the model itself is a good example of a complex system whose behavior is extremely hard to grasp without advanced tools. The manual tuning of all badly behaving control loops would be an extremely demanding task to carry out in a reasonable time.

In the case study, the production of pulp in a steady state operation was considered, i.e., changes of production rate or quality targets of the pulp were not simulated. This naturally fixes the results of the controller tuning to the chosen situation. However, in order to save time, a simple and short simulation sequence was applied. In the initial state of the test case, several control loops were behaving poorly. The level controllers in the impregnation and the digester vessels were oscillating heavily due to their inappropriate tuning. Due to an unsatisfactory level control, also the washing coefficient control failed to meet its targets. The two models applied in the process control were behaving even more detrimentally. The first one was used for predicting the total H factor value based on the digester top temperature measurement according to equations (5.9) and (5.10). There is a five hour time delay in the cooking process which makes this prediction essential for the process control. The other model was used for calculating the setpoint value for the H factor control based on the amount of applied alkali and the desired kappa number of the produced pulp. The calculation was based on the kappa model (5.11). Both models were giving strongly biased predictions, naturally causing serious problems to the process control. Due to the improper controller tuning, the process was continuously producing out of specification pulp.

5.2.3 Objectives of the case study

The cost function of the optimization was formulated according to equation (3.7) with $m = 6$ and $w_i = 1$, $\forall\ i = 1\ldots6$. All quality measures q_i were defined with the same mathematical expression,

$$q_i = \frac{1}{T}\sum_{t=1}^{T}\left|r_i(t) - y_i(t)\right|, \tag{5.12}$$

in which $i = 1\ldots6$, t is the time series sample index, T is the length of the simulated time series, y_i is the measured process variable and r_i is the setpoint value for y_i (except for $i = 6$, for which y_i is the predicted H factor and r_i is the true H factor calculated from the digester temperature profiles). The six variables considered in the quality measure calculation were:

y_1: kappa number of the cooked pulp,
y_2: washing coefficient,
y_3: liquor level in the digester,

y_4: chip level in the digester,
y_5: chip level in the impregnation vessel, and
y_6: H factor prediction.

The tuning involved seven PI controllers and the two models described above. Altogether n = 20 parameters were assigned to decision variables. The PI controllers were responsible for regulation of the chip and liquor levels, washing coefficient, production rate, H factor, and the digester steam chamber temperature.

The number of data points in each step of the global iteration varied between 40 and 100, the average amount being k = 77. About one fifth of data was always reserved to model validation and selecting the best latent dimension N for the model. Later it was discovered that typically around $k = 40$ samples were enough to obtain usable models (i.e., models that predicted the gradient direction correctly). The local models were calculated with *Canonical Correlation* based regression (CCR, see [34]) which gave slightly better results than the PLS models. The parameters were sampled from a multivariate Gaussian distribution with a fixed standard deviation, σ_θ = 3 % of the numerical (absolute) values of the parameters.

A simple IRT version with gradient descent update method and a constant step size γ was used in the optimization according to equation (3.9). Altogether, K = 45 global iteration steps were taken in the case study. A rather large number of global optimization steps were needed since the initial performance was extremely poor. Moreover, a somewhat crude version of the IRT method was applied in this case study. The main emphasis, however, was on studying how the large and complex system like the considered pulping process could be managed as a whole in the performance optimization.

The simulation sequences used in the quality measure calculations were T = 8h long and they were run about 25 times faster than real time (with a 1.67 GHz processor and 512 MB RAM). For example, it took about 12 hours to simulate the data required for one global step with k = 40.

5.2.4 Results

The IRT method succeeded in improving the process performance regarding all six quality measures (see Figure 14). For example, the deviation of kappa number from its target value, q_1, diminished from about 6 to 3 units, and the absolute value of the H factor prediction error, q_6, was reduced from the initial value q_6 = 1000 to less than 200 units. For softwood pulp, the kappa number of the produced pulp remains acceptable if it stays within ±2-3 units range from the target value. In that sense, the control is not yet excellent although the tuning succeeded in improving the performance notably. Based on the results it seems that the performance of the system could be improved further by continuing the IRT tuning. The tuning was left unfinished simply due to lack of time. The quality measure values include some stochastic variation and, therefore, the trends are not monotonically descending. Occasional "outliers" can be perceived in the values of the quality measures during the global iteration steps 15-20. The causes of these abnormal observations are discussed below. Furthermore, it can be seen from Figure 14 that the values of q_3 start to grow, meaning that targets start to become gradually contradictory to each other.

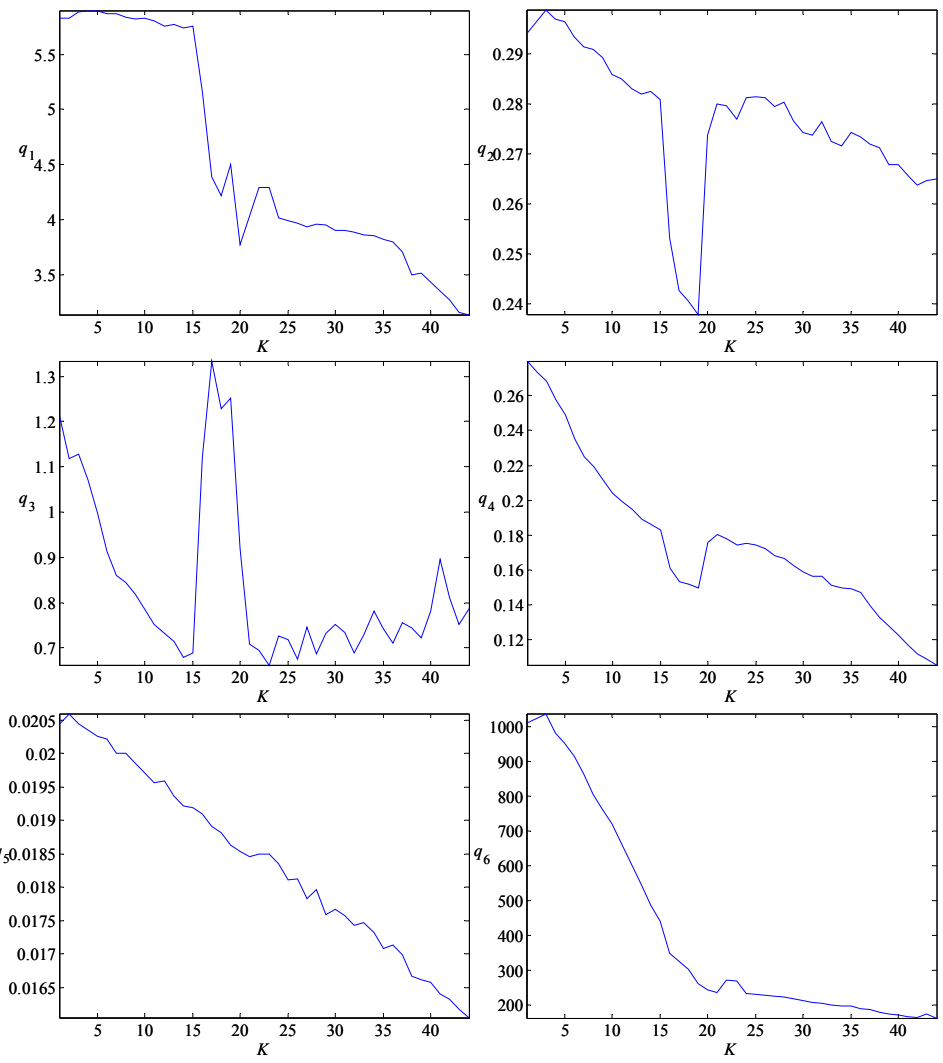

Figure 14. The quality measure values, q_i, i = 1…6, in the global optimization steps, K = 1…45.

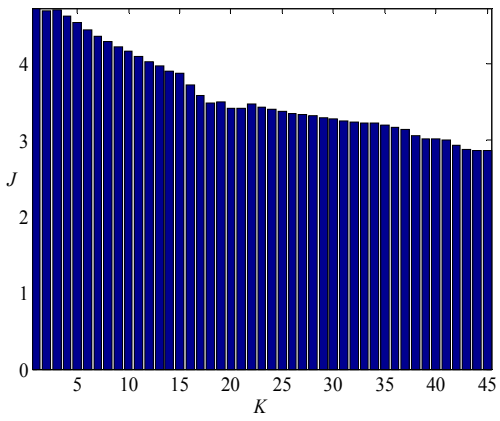

Figure 15. The values of the cost function, J, during the iterative optimization, K = 45.

The success of the optimization can be followed also from the values of J (see Figure 15). It can be seen how the conflicting targets finally start to slow down the tuning procedure. If the objectives are not met at the end, the weighting of q_i has to be

reconsidered. This reflects how the IRT method transforms the tuning of the individual lower level control parameters into adjustment of quality measure weights on the higher level. Examples of the improved performance are presented in Figure 16. The fluctuation of the kappa number has stopped and its deviation from the setpoint has become smaller, variance of the digester liquor level has decreased and the H factor prediction has improved tremendously (although the variance of the prediction has increased a little).

Any outlier detection method was not applied in the case study, which would have been beneficial. Most probably it would have alarmed about the heavily nonlinear cost function. The distributions of the quality measure data projected on the plane spanned by two most significant principal components in global steps $K = 18$ and $K = 30$ are shown in Figure 17. Obviously, the distribution in the left figure does not fulfil the Gaussianity assumption. The same can be seen also from Figure 18 in which the empirical cumulative distribution functions (ECDF) of the distributions are compared to the theoretical CDF of the (0,1) Gaussian distribution. This comparison is known as the Kolmogorov-Smirnov test [76]. It is evident that at some points in the parameter space the cost function may be strongly nonlinear which shows as peculiar spikes in Figure 14. Therefore, Gaussianity testing is recommended to determine the appropriate standard deviation σ_θ for the local variation.

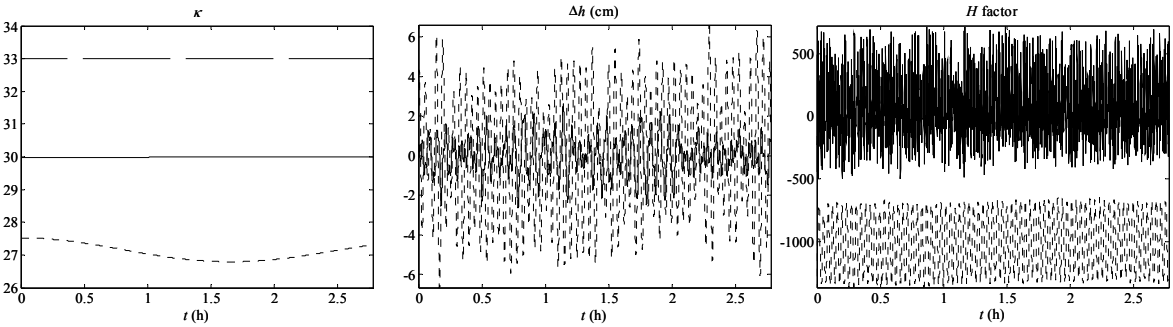

Figure 16. Control performance of kappa number κ ($\kappa_{sp} = 33$ marked with dashed line), digester liquor level deviation Δh and H factor prediction error, initial performance $K = 1$ (dotted) and $K = 45$ (solid).

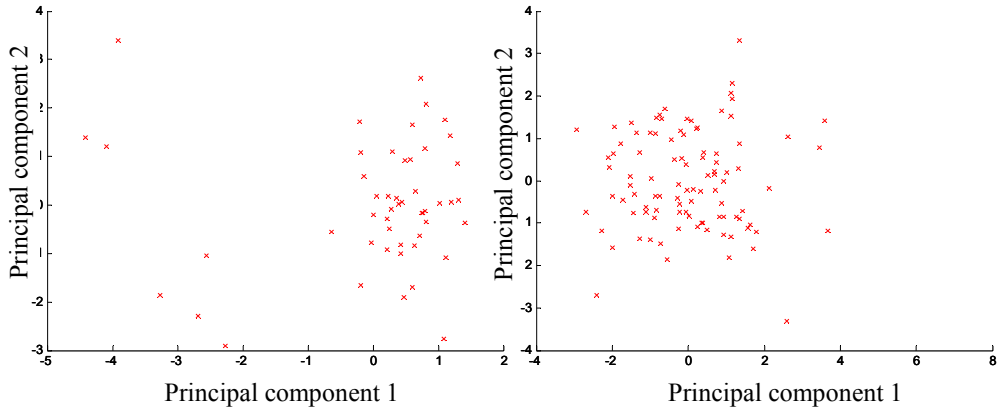

Figure 17. Distributions of q values projected to the plane spanned by the two major principal components, $K = 18$ (left) and $K = 30$ (right).

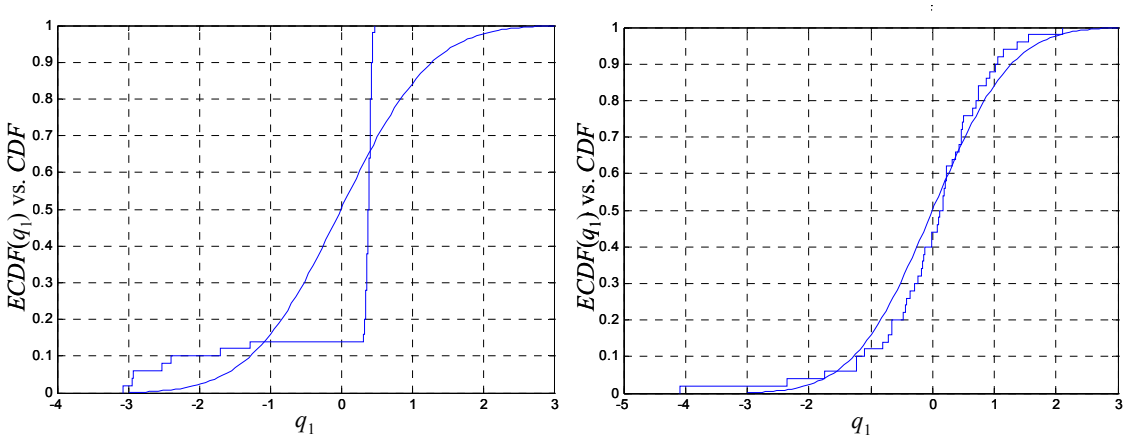

Figure 18. ECDF of q_1 vs. theoretical CDF of the (0,1) Gaussian distribution, K = 18 (left) and K = 30 (right).

The actual parameter values are not important here, but some examples of the update trends are given in Figure 19. All parameter values did not necessarily evolve consistently from K = 1 to K = 45 but some of them might, for instance, change their direction at some point of the optimization procedure. For example, the value of θ_5 changed more or less consistently to the new value and finally settled there. The parameters θ_8 and θ_{10} instead evolved in spurts, and no sign of settling down can be perceived in the values of θ_{13}. These results strengthen the assumption that the global cost function is nonlinear, but smooth enough to be approximated with locally linear models.

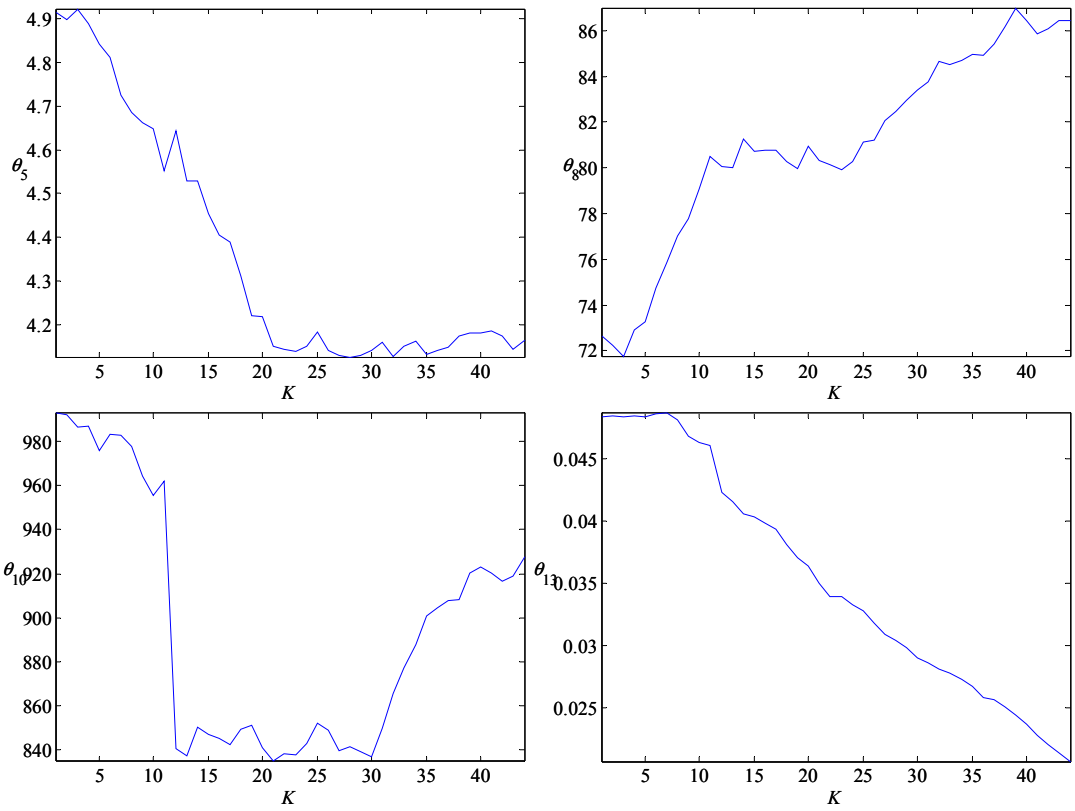

Figure 19. Four examples of parameter evolution trends during K = 1…45.

In conclusion, grasping the general view of a large and complex system, like an industrial process and its control system, is a demanding task. The underlying interdependencies are difficult to comprehend as the size of a system increases. Advanced multivariate statistical methods are required to capture the emerging higher abstraction level concepts.

Obviously, the IRT method makes it possible to optimize the performance of large systems, and helps the domain area experts to refine their intuition concerning the process and the goals of its performance. As long as the distribution of the quality measures stays close to Gaussian, the IRT method proceeds convincingly. Therefore, defining q should be done carefully. Departures from Gaussianity can be due to the nonlinear behavior of the quality measure or too large local variation of the parameters, or both. Despite the severe nonlinearities of q_i the values of J, the weighted sum of q_i, in Figure 15 show that overall performance of the control system is improved consistently.

Since running a tuning procedure using IRT is a random process in itself it would have been interesting to see how the obtained results change if the same procedure was repeated. And further, how the results change if, for example, different initial values were applied.

5.2.5 Global nonlinearity vs. local linearity

With the Veitsiluoto simulation model some tests were run concerning the local linearity assumption and extrapolation beyond $\Re_S$ in the case of a nonlinear cost function. Two sets of data from global steps K = 21 and K = 40 were used in the study. The later represents a situation in which the local linearity was well justified and the first one demonstrates a situation of heavily nonlinear target functions.

Bera-Jarque and Kolmogorov-Smirnov tests were used for studying the marginal Gaussianity of the data distributions (see [76] for more sophisticated methods). According to the assumption, the data set from step K = 40 passed both normality tests. The other data set seemed to be heavily non-Gaussian, as expected, which is illustrated in Figure 20. Only q_5 and q_6 passed both Gaussianity tests with a significance level $\alpha = 0.05$.

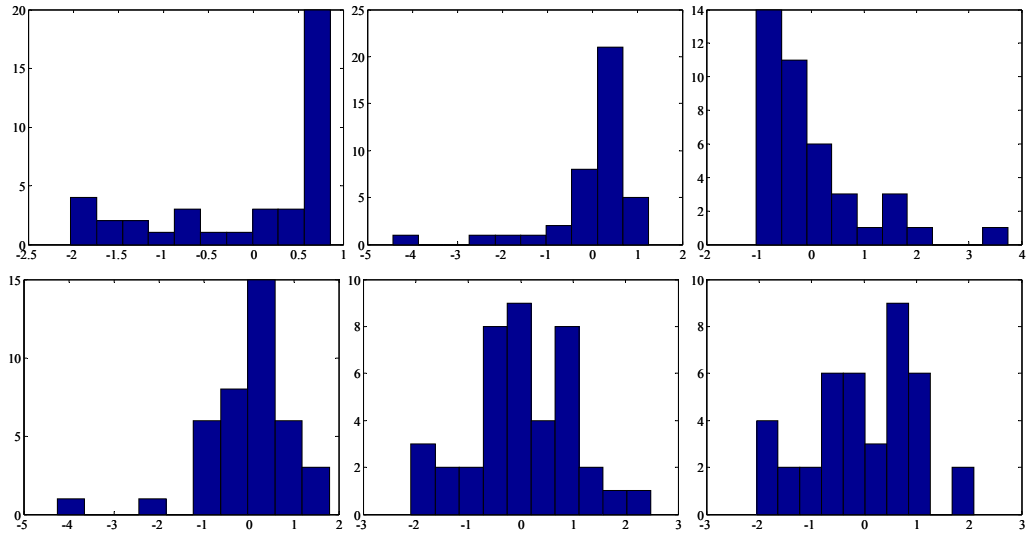

Figure 20. Histograms of q_i distributions, i = 1...6, of the K = 21 global iteration step (from top left to bottom right).

The extrapolation capability of the estimated models based on these two data sets were tested with different values of step length coefficient, $\gamma \in$ {0.2, 0.4, 0.6, 0.8, 1, 2, 3, 4, 5, 10}, see Figure 21 and Figure 22. The corresponding observed cost function values are presented in Figure 23 and Figure 24. Obviously, extrapolation far beyond $\Re_S$ may result in problems even when $\Re_S$ is small enough and the model works well locally. From Figure 22 it can be easily seen that $\Re_S$ is too large related to the shape of the cost function. Nonlinearity appears already inside $\Re_S$.

In conclusion, studying the distribution of the observed quality measure values seems to be beneficial. It can effectively help in discovering the situations in which human decision making is needed most acutely with respect to the size of $\Re_S$ and weighting of q_i.

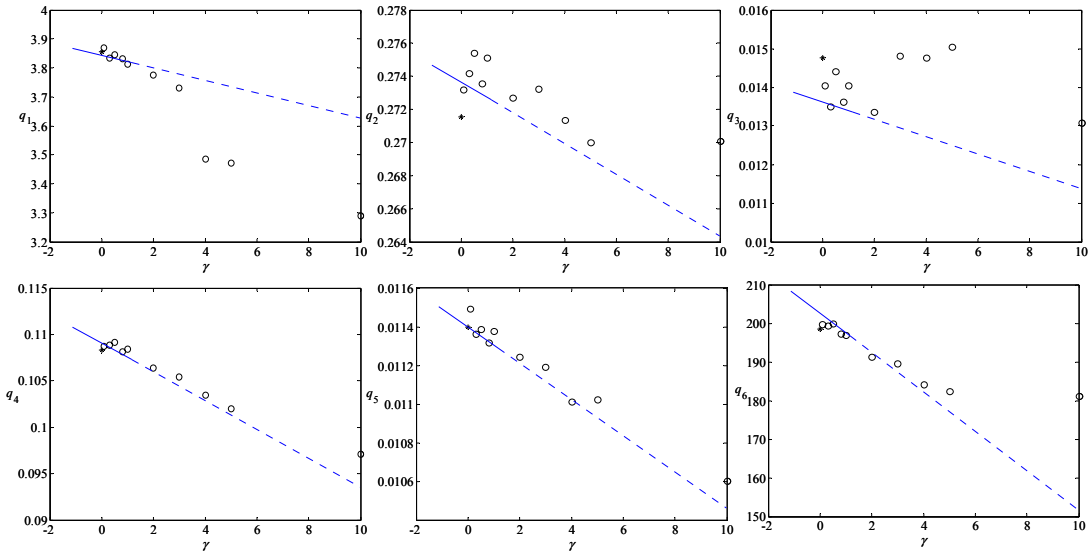

Figure 21. Extrapolated values of q_i (blue dashed line) beyond $\Re_S$ (blue solid line) from the data set $K = 40$ compared to observed values (black circles) as a function of the step length coefficient γ.

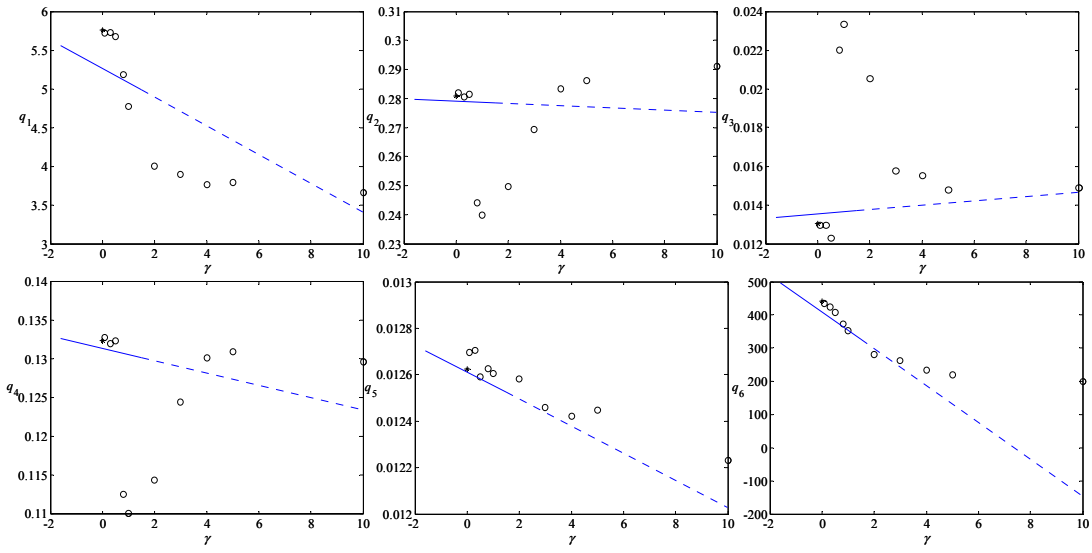

Figure 22. Extrapolated values of q_i (blue dashed line) beyond $\Re_S$ (blue solid line) from the data set $K = 21$ compared to observed values (black circles) as a function of the step length coefficient γ.

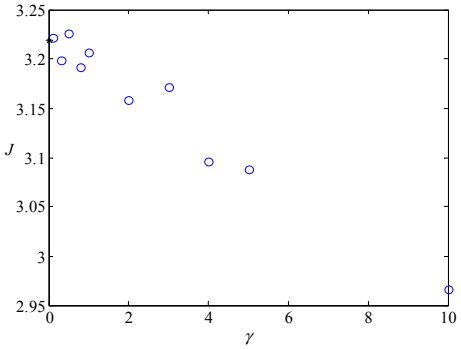

Figure 23. Observed cost function values *J* corresponding to the extrapolated parameter updates as a function of the step length coefficient γ after the global step *K* = 40.

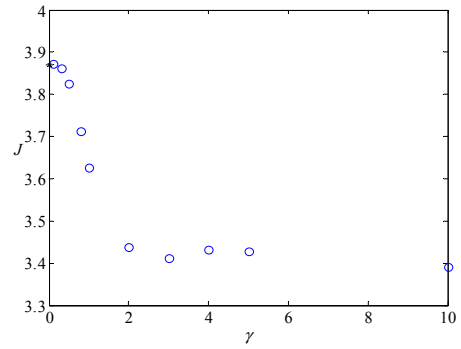

Figure 24. Observed cost function values *J* corresponding to the extrapolated parameter updates as a function of the step length coefficient γ after the global step *K* = 21.

5.3 Heat exchanger case study

The aim of this case study was to test the tuning of simulation model parameters in practice with a model that has some real industrial relevance. The results were originally published in [23].

5.3.1 Process and model description

The model (presented in Figure 25) comprises two intermediate counter-current multitubular heat exchangers, and it is a part of the overall model of the Olkiluoto 2 nuclear power plant. The considered part of the process has been replaced with another solution on the existing plant but the model works well in research use. In the heat exchanger system, the high pressure heating steam (around 50 bar) flows inside horizontal tubes and the heated steam flows on the shell side (in the following also referred as the outside). The heating steam is split into two parallel flows whereas the heated steam flows through both heat exchangers one after another. In the following, the lower heat exchanger in Figure 25 will be referred to as the first exchanger, the upper being called then the second heat exchanger. The goal of the tuning was to improve the accuracy of the simulation results by tuning the model parameters related to the heat transfer from the steam flow on the tube side to the shell side.

Since no measurement data was available from the actual power plant the simulations were compared to "simulated measurements", i.e., a new version of the same model with improved steam condensation and gas convective heat transfer calculation principles were used to produce the measurement data. This improved version had been validated with process measurements and it was proven to give more accurate simulation results than the old one. Obviously, this kind of a case study is an artificial example of using process measurement data for model parameter tuning since two relatively small simulation models are easily run with equivalent input signals and discrepancies between the results are only due to the differing model structures and insufficient tuning of model parameters. In reality, comparing simulation results to measurements is far more complicated as the size of the system and the number of the signals grows. Missing and erroneous measurements are also fairly common inconveniences when playing with real data. However, this case study still enables one to test the basic idea of applying the IRT method to model parameter tuning.

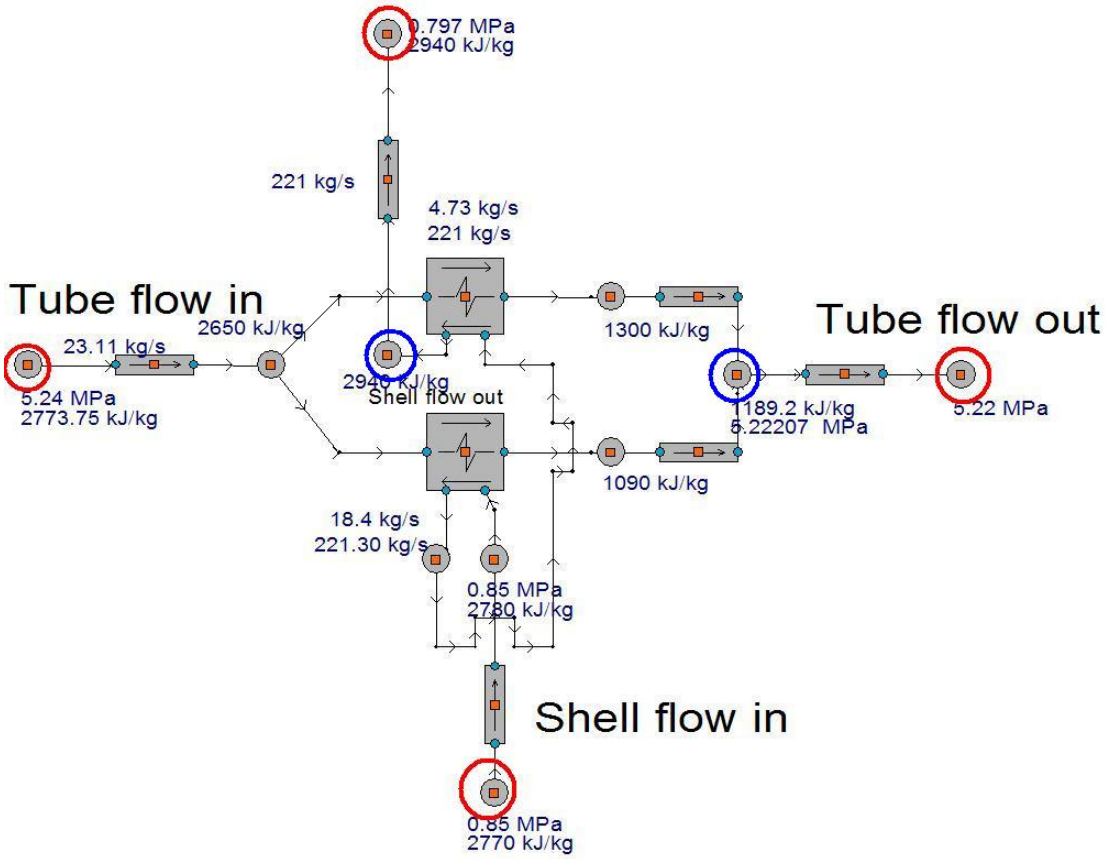

Figure 25. Olkiluoto intermediate heat exchangers modeled with Apros simulation software. The boundary points of model are circulated with red and the nodes from which the quality measures are evaluated are circulated with blue.

Both of the heat exchanger modules in the simulation model are divided into eight calculation volumes (nodes) along the tube flow. Based on the surrounding temperature, properties of the fluid and process equipment, as well as flow characteristics Apros calculates values for the heat transfer coefficients in each node both for the tube and shell side surfaces. Apros also takes into account the material and the geometry of the heat exchangers to calculate the heat conduction from one side to another. The overall heat conductance in one node, P_i, from the heating flow to the heated flow can be expressed as,

$$P_i = \frac{1}{\frac{1}{\eta_{\text{out},i}\alpha_{\text{out},i}} + \frac{h}{\beta_i} + \frac{1}{\eta_{\text{in},i}\alpha_{\text{in},i}}}, \tag{5.13}$$

where α_{in} and α_{out} are the inner and outer surface heat transfer coefficients, h is the wall thickness and β is the heat conduction through the tube wall. In addition, both of the interfacial heat transfer coefficients are equipped with multiplicative efficiency coefficients η in the simulation model, which can be used to fine-tune the accuracy of the model.

5.3.2 Objectives of the case study

In the applied simulation sequence the power production level of the power plant was lowered from 100% production to 90%. This was done by ramping the pressure and enthalpy values of the boundary points of the simulation model (see Figure 25) according to the process data obtained from the power plant. Some noise was also added on the boundary points of the model. The length of the resulting simulation sequence was 6 minutes and it took about 10 seconds in real time to run one simulation on an ordinary office computer.

Root Mean Squared Error (RMSE) criteria of the output enthalpy and the output mass flow were used as quality measures. They were defined both for the tube side and the shell side flows resulting in four quality measures altogether. The quality measure values were evaluated from the nodes marked with blue in Figure 25.

The efficiency parameters η of the interfacial heat transfer coefficients in the eight calculation nodes of both heat exchangers were chosen as decision variables resulting in 32 decision variables altogether. Initial values $\theta_i = \eta_i = 1$, $\forall i$, were used meaning that the heat transfer coefficients calculated by Apros were not corrected with the efficiency parameters.

The tuning was repeated several times with different versions of IRT (for example normal vs. recursive PLS, different sample sizes, sampling distributions). Tremendous differences were not perceived between sensible candidates and results are presented only from the following setting. Each global optimization step consisted of $k = 60$ data points and since one simulation run took about ten seconds, completing the whole tuning procedure lasted about three and a half hours. A local model was computed using the semi-recursive PLS modeling presented in Section 0 with $\mu = 0.9$ and $N = 3$. The update step was computed using $w_i = 1$, $\forall i$. Gaussian sampling distribution with $\sigma_i = 0.01$ for each θ_i was applied. The step length coefficient γ was chosen such that,

$$\begin{aligned} \|d\| &= \|-\gamma \cdot \nabla J\| \\ &= \max_{\kappa} \left(\|\Delta\theta(\kappa)\| \right), 1 \le \kappa \le k. \end{aligned} \tag{5.14}$$

Equation (5.14) suggests that the update steps were truncated to the same length with the largest parameter perturbation in the current local iteration.

5.3.3 Results

The progress of the optimization can be examined either from the cost function (Figure 26) or the quality measure values (Figure 27). Both figures reveal that the optimization reaches the local optimum with about $K = 10$ global steps after which it is unable to find a direction in the decision space that would yield an improvement.

The accuracy of the local linear model can be evaluated by comparing the obtained quality measure values to the estimates calculated with the local model. In Figure 28 scatter plots on q_i against the estimates of q_i are presented from the data of global optimization step $K = 1$. The accuracy of the linear model remains approximately the same in every optimization step, $K = 1$ - 20. The correlation of q_3 with its estimate appears to be stronger than that of q_1, q_2 and q_4. Obviously, the reason for this is the lower noise level on the heated steam output enthalpy signal based on which q_3 is calculated.

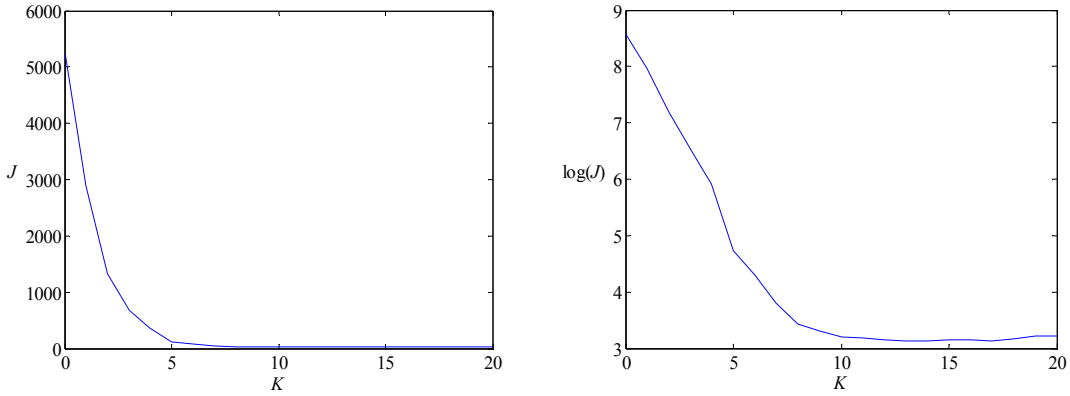

Figure 26. The cost function values during $K = 20$ global optimization steps.

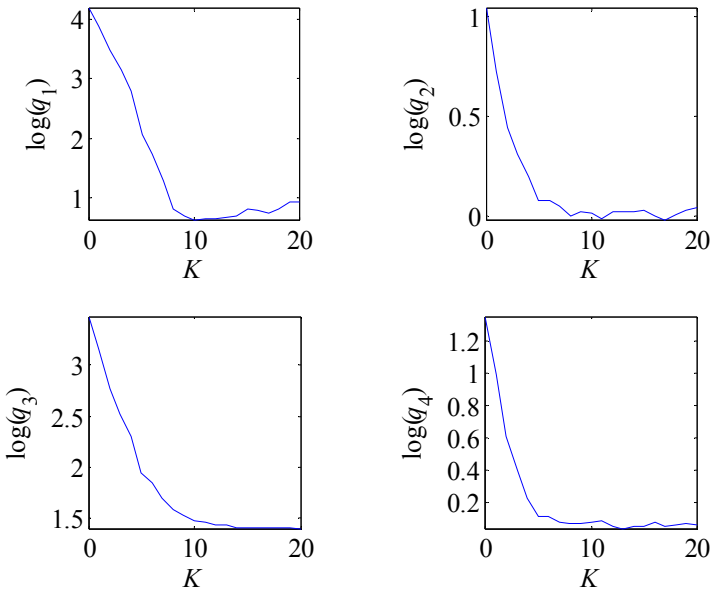

Figure 27. Quality measure values q_1 - q_4 during $K = 20$ global optimization steps.

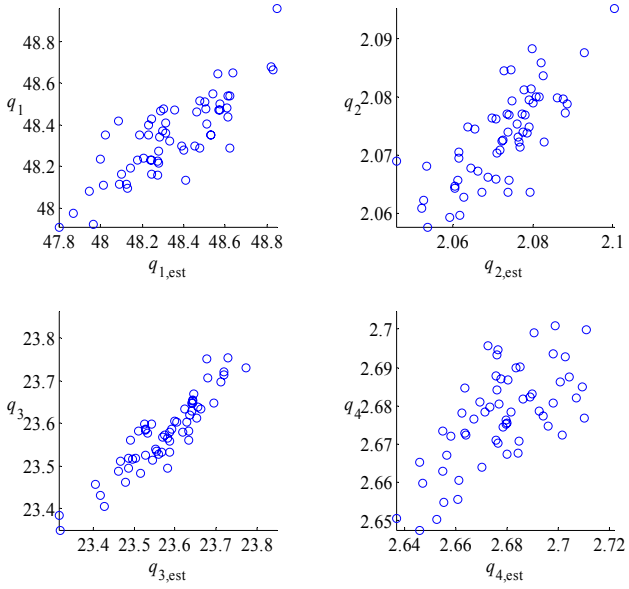

Figure 28. Comparison of the observed and the estimated values of the quality measures in $K = 1$ global optimization step.

Figure 29 presents the parameter trends during 20 global iteration steps. The initial values of the parameters were $\theta_i = 1$, $\forall i$. As can be seen, the tuning had a stronger effect on the shell side coefficients and especially on the shell side coefficients of the first heat exchanger. This kind of a result was expected already beforehand since the effect of the outer surface heat transfer coefficients on the total heat conductance is known to be an order of magnitude greater (at this particular setting) than that of the inner surface coefficients. Also, the first heat exchanger is responsible for the most of the energy transfer between the two steam flows, since the temperature difference over its tube walls is larger than in the second heat exchanger, which improves the heat conduction through the tube walls.

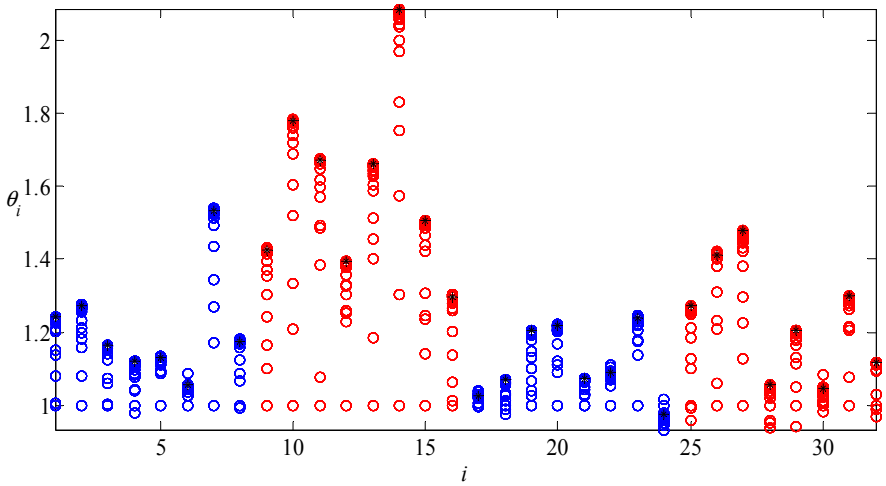

Figure 29. Parameter values θ_1 - θ_{32} during 20 optimization steps. Parameters θ_1 - θ_{16} are efficiency coefficients of the first heat exchanger and θ_{17} - θ_{32} are efficiency coefficients of the second heat exchanger, inner and outer surface coefficients are marked with blue and red, respectively, and the final values are denoted with black asterisks.

The improvement of the cost function and the quality measure values stop after the global step K = 10 since the quality measures start to compete with each others at that point. This can be shown, for example, by studying the angles between the columns of the matrix F (see Figure 30) which can be interpreted as gradients with respect to each individual quality measure. The conflict between q_1 and q_3 is especially eye-catching – the improvement of one of them necessarily results in the deterioration of the other. Initially, it is possible to find such directions from the n-dimensional parameter space that yield improvements with respect to all quality measures (in other words, directions that yield in *Pareto improvements*). When K > 10 the individual gradients become orthogonal to each others on average, meaning that the optimization can be continued only as a decision making problem using the weightings of the quality measures as decision variables.

In Figure 31 - Figure 34 simulation results of the four considered output signals of the model are compared to the measured signals during the optimization procedure. The bias from the simulation results has vanished after K = 5 - 10 optimization steps, except for the heated steam enthalpy which differs from the measurement signal still after K = 20 steps. Its estimation accuracy competes with heating steam enthalpy. The bias originates from the differences between the model structures and, therefore, cannot be completely overcome by parameter tuning.

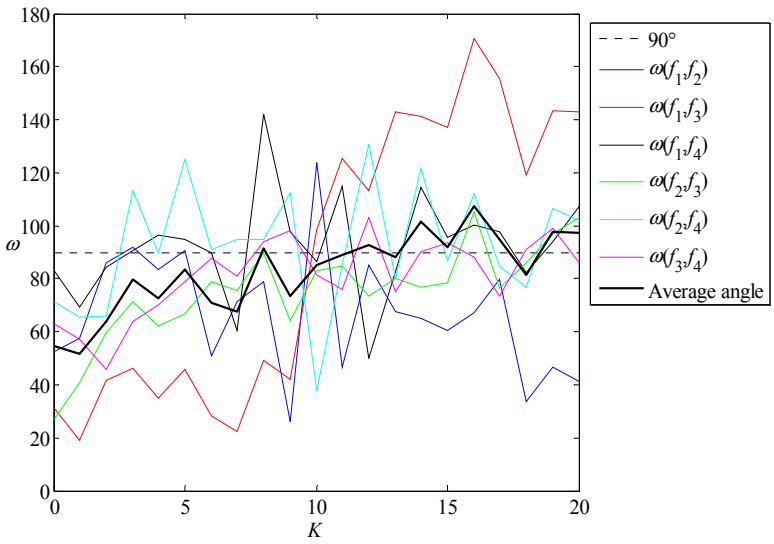

Figure 30. The angles between columns of mapping matrix F during K = 1…20 global optimization steps. The average angle between columns of F is drawn with the thick black line.

The accuracy of the eigenproblem oriented PLS regression and MLR models were compared with the data of this case study. Comparison was based on the mean RMSE criterion of the four output signals and it was calculated using Leave-one-out cross-validation. It turned out that the PLS outperformed MLR substantially. Due to the heavily collinear decision variables, the MLR model becomes unreliable. (See also the next case study where different conventional PLS algorithms are compared to the eigenproblem oriented PLS regression and MLR models).

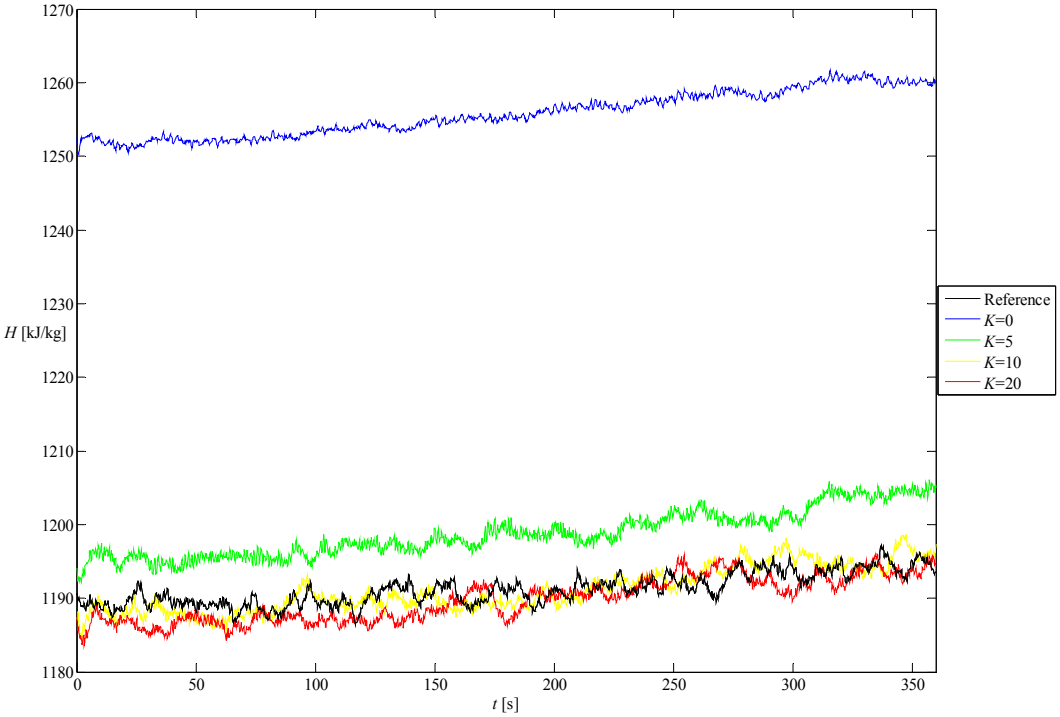

Figure 31. Enthalpy of the heating steam flowing out of the heat exchanger system during a decrease of power production from 100% to 90% level.

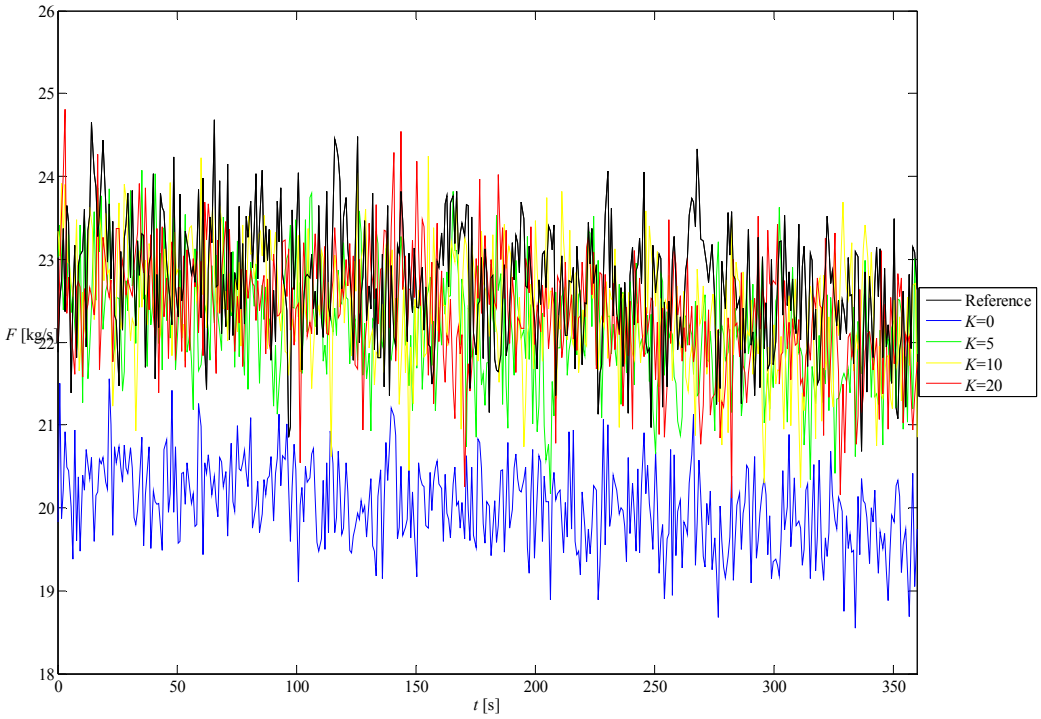

Figure 32. Mass flow of the heating steam flowing out of the heat exchanger system during a decrease of power production from 100% to 90% level.

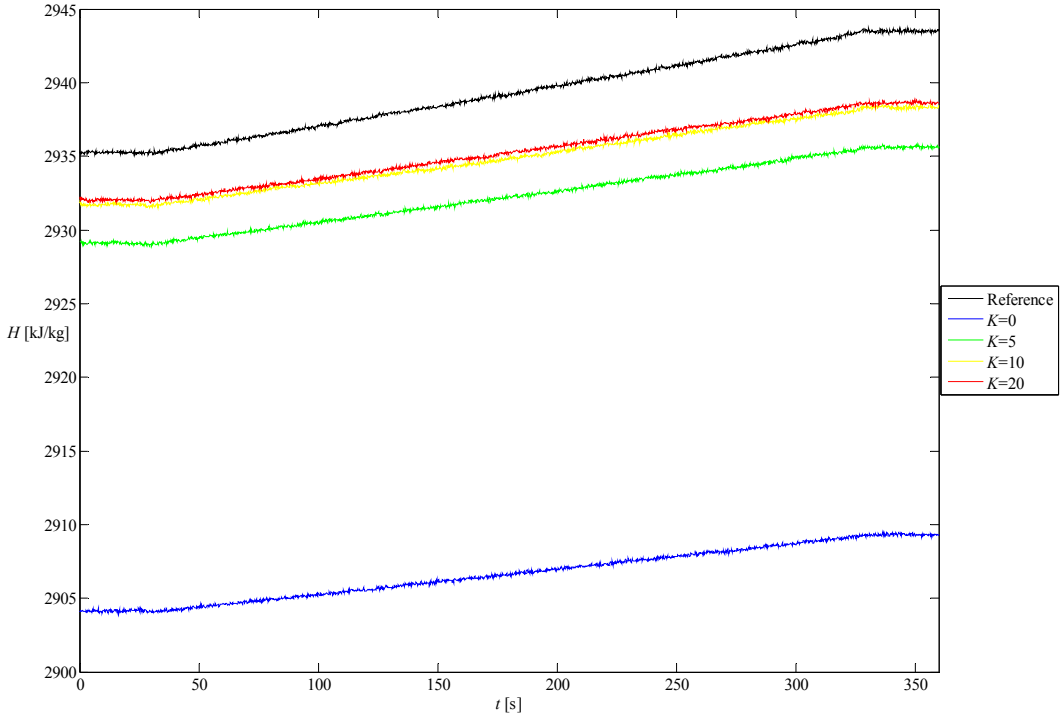

Figure 33. Enthalpy of the heated steam flowing out of the heat exchanger system during a decrease of power production from 100% to 90% level.

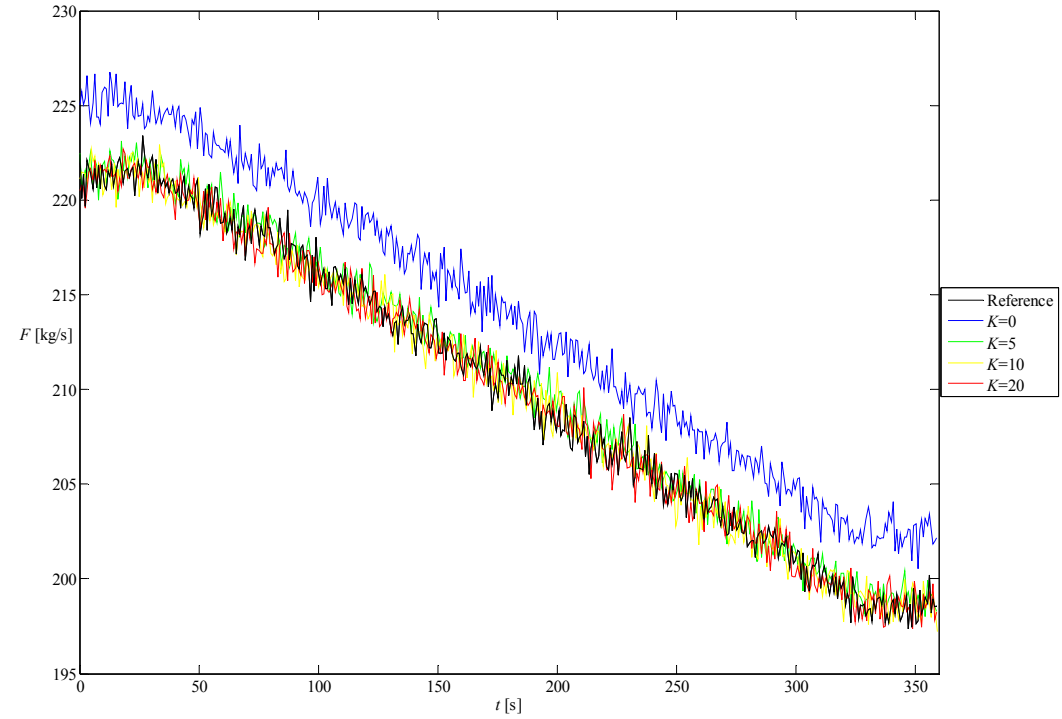

Figure 34. Mass flow of the heated steam flowing out of the heat exchanger system during a decrease of power production from 100% to 90% level.

5.3.4 Discussion

The results converge without any interruptions to the optimum in this case study which suggests that the problem was formulated successfully – the quality measures were sensibly formulated and the local parameter sampling area was practical. The linear model predicts the dependency between θ and q relatively well if estimated using PLS. In case of a nonlinear target functions, the scatter plots similar to those presented in Figure 28 would reveal the nonlinearity if only the noise level is moderate enough. In the local optimum the conflicting targets make the angles between the columns of F matrix orthogonal or even contradictory which ends the optimization.

The different accuracies of variable estimates (or the strength of the correlation between observed and estimated outputs) are explained by the different signal-to-noise ratios in the data. In practice, it is difficult to influence the observed noise in the time series signals. By increasing the input variation the ratio can be improved as long as the local linearity assumption holds.

The IRT method succeeds to tune the set of parameters in a sensible way such that the simulation results become more accurate as compared to the measurement data. This can be seen either from the quality measure trends or from the simulation results. Only the enthalpy values of the heated steam flow (corresponding to q_3) remain slightly biased at the end of the optimization as compared to the measurements. In addition to the bias in the signals, there is also a small difference in the slopes in Figure 33. This highlights the differences of the two model structures (the reference model and the one whose parameters were tuned) that are used in this case study. By merely tuning the model parameters one cannot obtain equal simulation results.

The IRT method also seems to be able to distinguish the significant decision variables from those of having only a negligible effect on the outputs. The tuning mainly concentrates on the significant ones (the outer surface heat transfer coefficients) changing mainly their values during the optimization. The strongest dependencies among the variables become visible only with a rather small number of data samples, whereas the weaker correlations need much more data before they can be detected. Since the tuning was repeated several times, it could be noticed that the end results vary with respect to individual values of the decision variables on different tuning runs. The performance of the system and the "structure" on the final values of decision variables, however, remained essentially the same.

5.4 Screening department of CTMP plant

Another case study about model parameter tuning using the IRT method was conducted with an Apros model of the screening department of Kaukopää CTMP (chemi-thermomechanical pulp) plant. The target of this study was to test the scalability of the IRT method with an extremely large model comprising dozens of tuning parameters and important output signals.

5.4.1 Process and model description

In CTMP pulping the goal is to combine the advantages of chemical and mechanical pulping – the higher fiber length and strength of chemical pulp and the better optical

properties and higher yield of mechanical pulping. CTMP pulp is used, for example, in production of tissue and liquid packaging board products. The process consists of steaming, chemical impregnation, heating and refining stages. The chemical treatment of wood chips reduces the energy demand of the refining process. In the refining process, chips are broken down and screening is needed to separate impurities and larger wooden particles that require more refining from the fibrous material. Centrifugal cleaners separate wooden material from impurities like sand based on the density difference of particles after which refining of wooden particles is repeated.

The model of the screening department includes three screens and a reject handling section. The feed stock flow is split between the two primary screens (Figure 35) and the reject flows from both of them are collected into a reject vessel and fed to a reject screen (Figure 36). The reject flow from the reject screen is first led through a section of centrifugal cleaners where the impurities from the pulp are removed. After that, the reject is refined in order to break up the coarse fraction that contains shives and large fiber bundles. The reject handling part of the model was implemented using a lower accuracy level (referring to the different modeling paradigms of Apros) and, therefore, pressure measurements were available only from the screening and centrifugal cleaner section. The control system was reconstructed in the simulation model using the blueprints of the existing implementation and the correct control parameter values were adopted from the plant data.

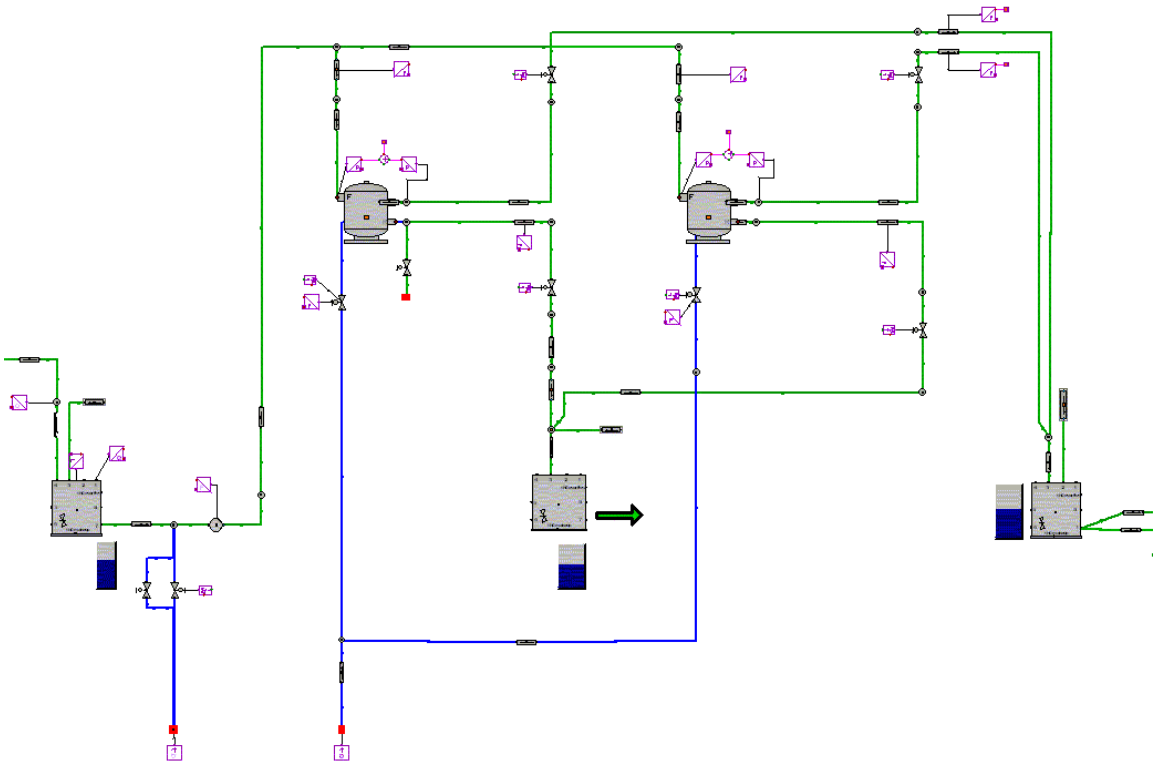

Figure 35. Screening department of a CTMP plant (components from left to right): feedstock tank, screen 1, reject vessel, screen 2, and tank for screened pulp.

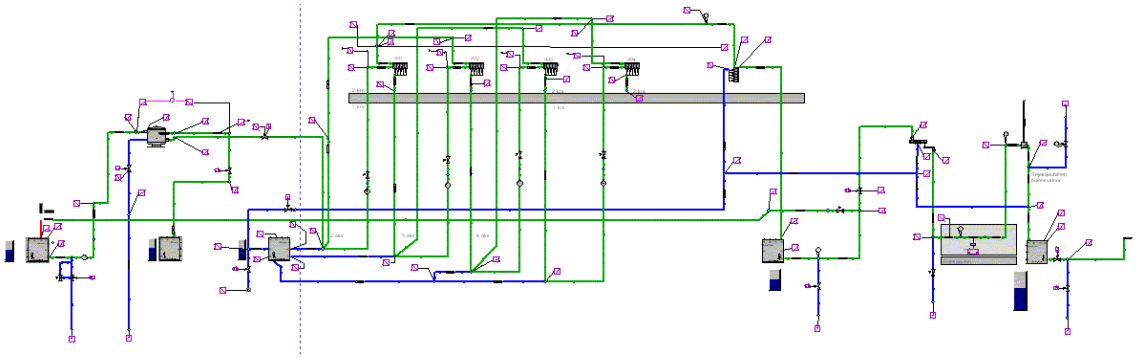

Figure 36. Reject handling department of a CTMP plant (components from left to right): reject vessel, reject screen, tank for screened pulp, feeding tank of the centrifugal cleaner section, cleaner section, intermediate storage tank, and reject refiner.

Three separate operating points were distinguished based on plant personnel interviews – high production rate with two screens in operation and low production rate with either one or two screens in operation. The target of this tuning case was to reconstruct the first of them as precisely as possible. A period of data describing the situation without external disturbances or unusual operator interventions was carefully chosen from the historical data recorded at the process plant. Constant input and reference signals corresponding to the mean values of the time series signals from the chosen data period were used in order to get the estimated outputs to the correct average level.

Tuning focused on 86 model parameters including the nominal degree of opening, nominal pressure difference and nominal mass flow of the control and shut-off valves, the flow resistances of pipe components and accept ratios of screens for different fiber length fractions. The accept ratios of each screen were lumped together in order to maintain their relative proportions (that were based on an expert opinion) and, therefore, 77 parameters were finally addressed to the IRT method. The parameter values that were obtained from a reliable source, for instance, the characteristic curves of pumps and valves given by the manufacturers, were not considered in tuning. Additionally, parameters having no effect on the dynamic balance, such as the driving time of valves, were excluded from tuning.

The tuning targets were expressed using 61 quality measures. Each of them measured the (absolute) difference between the measured and simulated expectation value of a process variable. The signals consisted of flow, consistency, level, pressure or pressure difference measurements and controller output signals. For example, freeness, temperature, pH and other measurements describing the chemical content of the fiber-water suspension were excluded from the model and, therefore, also from the tuning. The target accuracy was determined for each signal (one for each SI unit, to be exact). The inverse of the target accuracies for each unit were used as multiplicative weights in the cost function in order to make the measurements in different SI units commensurable. The target accuracies for flow, consistency, level, pressure and control signals were chosen as 1 kg/s, 0,2 % (units of consistency), 1 % (per cents of the measurement range), 2 kPa and 1 %, respectively, resulting in weight values 1, 5, 1, 0.5 and 1.

5.4.2 Results

The first local iteration was carried out as an initial analysis consisting of $k = 500 \approx 6{,}5 \cdot n$ samples. In the later steps of global iteration the number of local iterations was $k = 150 \approx 2 \cdot n$. The shapes of the residual distributions were checked in order to detect departures from unimodality (and Gaussianity). Parameter variation was set initially to 3 per cents of the absolute values of the parameters. Based on the initial analysis also different PLS algorithms (see [8, 34]) were compared (see Figure 37). The eigenproblem oriented PLS formulation slightly outperforms the other PLS versions (modified Kernel algorithm being in practice as good) and the MLR regression model, achieving its best performance with $N = 34$ latent variables. The comparison was repeated with several smaller data sets after few global iteration steps. It turned out that the differences between the different PLS algorithms were relatively small (NIPALS algorithm typically giving the smallest error) as compared to the MLR regression that fell substantially behind when a scarce data set was used in modeling.

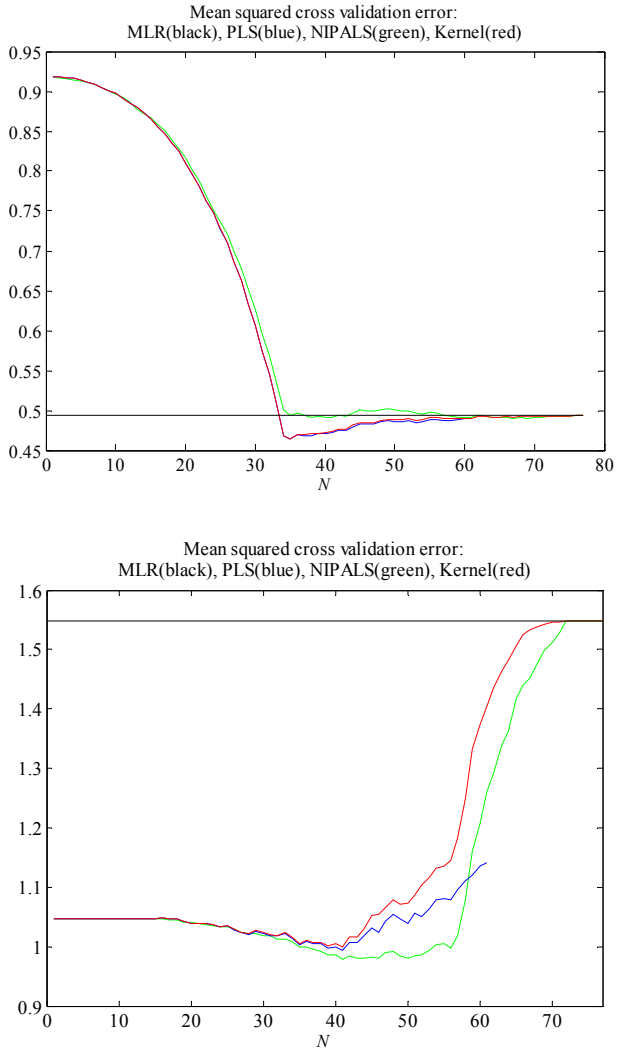

Figure 37. Cross validation MSE between observations and model predictions as a function of latent dimension *N*. The first global iteration step with $k = 500$ (above) vs. the global iteration step $K = 10$ with $k = 150$ (below). Abbreviation PLS refers to the eigenproblem oriented PLS formulation.

The progress of tuning was followed using trends of cost function and quality measure values. As can be seen from Figure 38, the tuning has not yet reached its target ($J/m = 1$, i.e., the error of each output estimate is one *commensurable unit* on the average) or any local optimum (not to speak of the theoretical optimum, $J/m = 0$) after $K = 27$ iterations. The optimization process was not finished since the main emphasis here was on the development of the Tuning Tool.

When the tuning results were analyzed afterwards, it was discovered that about half of the cost function reading comes from the pressure measurements of centrifugal cleaner section that have different interpretations in the model and in the data (absolute pressure values vs. pressure differences from a reference level). Hence the cost function values give an overly pessimistic view on the model accuracy. The values of Figure 38 would run approximately from 13 to 9 if the effect of the faulty pressure measurements was cleared. Further, it needs to be noticed that even though the target value is not necessarily achievable with the existing model structure, the trend in Figure 38 suggests that some improvement could still take place. Since the batch version of the IRT algorithm was applied some zigzag pattern can be seen in the cost function values.

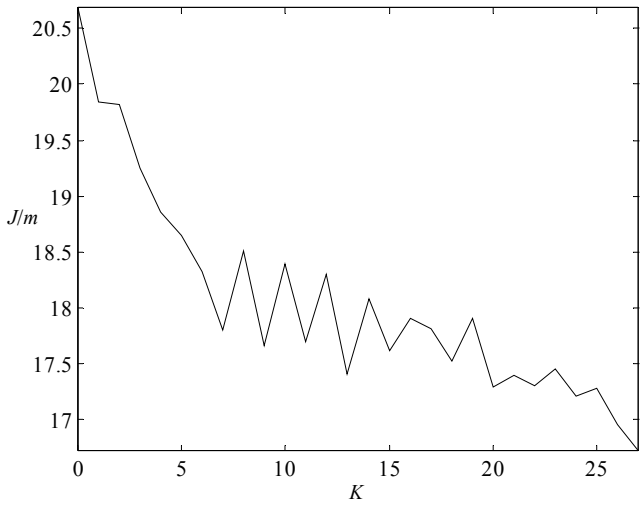

Figure 38. Mean deviation of q_i from the theoretical optimum ($q_i = 0$), i.e., J/m, as a function of global iteration index K.

From the 61 quality measures

- 12 were already initially in the optimum,
- 4 were close to the target values,
- 9 were successfully optimized during the 27 first iterations close to their target values and
- 19 were improved but the target values were not yet reached.

In other words, the values of 44 quality measures out of 61 were successfully optimized during the tuning procedure. Examples of these are given in Figure 39.

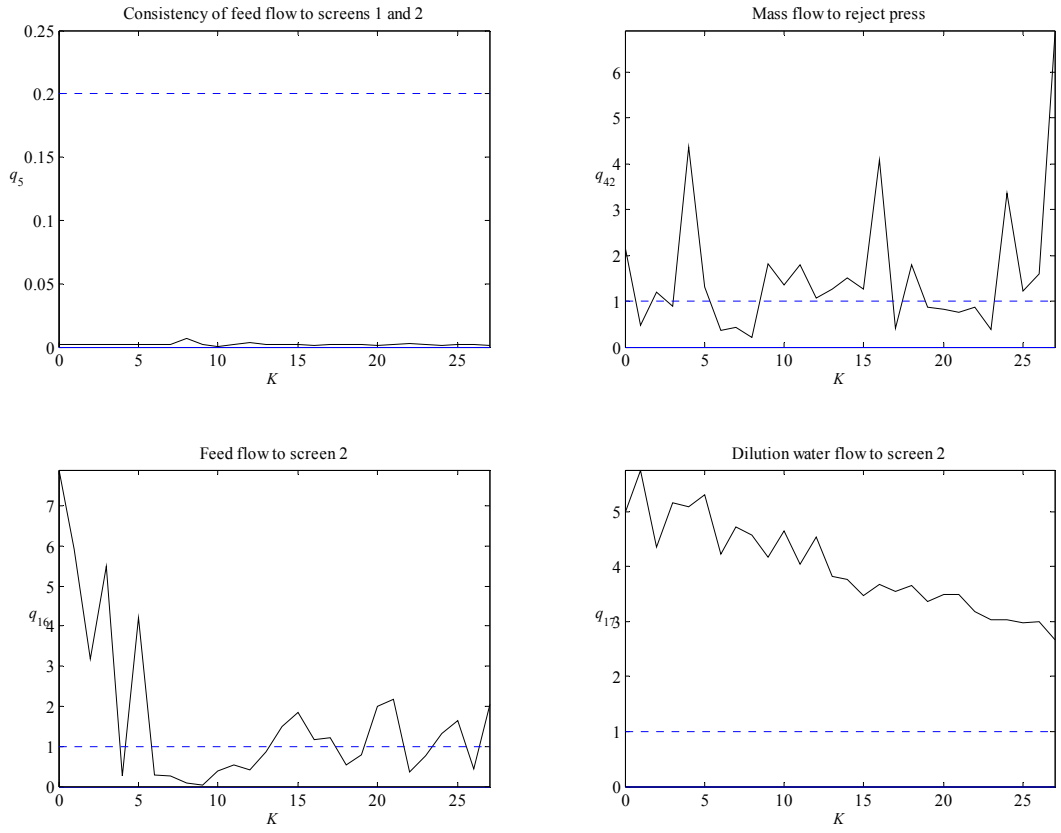

Figure 39. From top left: Quality measure value in the optimum, value is varying close to the target, value is successfully optimized to the target, value is decreasing but not yet in the target. Theoretical optimal value of quality measures is zero. Commensurable target value is marked with blue dashed line.

However, 17 quality measures presented some problems during the 27 iterations (see Figure 40):

- the values of 11 quality measures could not be affected significantly and
- the values of 6 quality measures deteriorated during tuning.

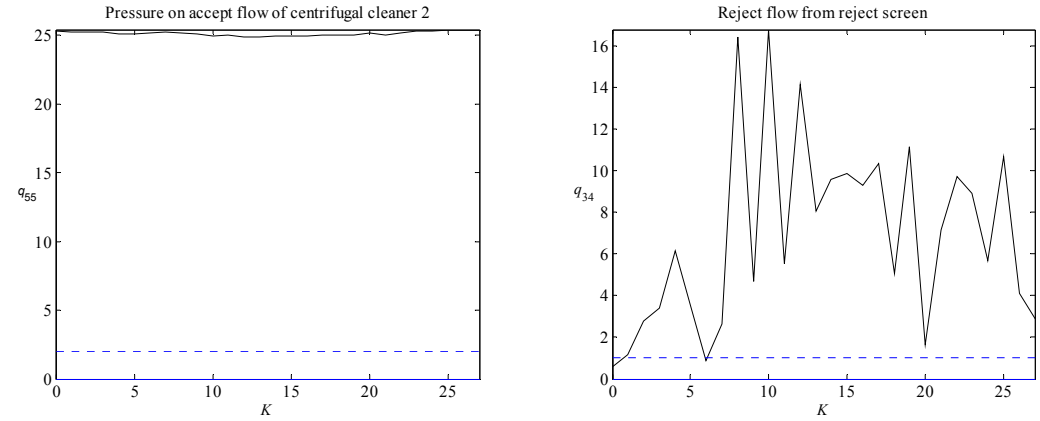

Figure 40. Bad examples: Tuning has had no effect on the quality measure values (left), the values have deteriorated during tuning (right).

5.4.3 Discussion

It is quite problematic to evaluate the success of an unfinished case study. Also when the number of targets rises this high capturing the big picture is non-trivial. What can be said based on the obtained results is that they speak strongly for a successful beginning. The trend of the cost function suggests that continuing the tuning would still improve the performance. Probably also the quality measures that initially showed no improvement would later respond to the tuning as the other quality measures arrive in the vicinity of the optimum. An example of this size also clearly points out the benefits from the latent variable based regression. As the number of variables grows and one needs to keep the sample sizes moderate at the same time, notably better models can be obtained using (any of) the PLS algorithms instead of MLR. Nothing general about the mutual order of superiority between different PLS algorithms can be concluded based on couple of examples, but obviously the differences are small in any case.

The case study showed that the numerical stability of the simulator plays an important role in successful application of the IRT method or any other data based parameter tuning technique. It also needs to be kept in mind that manual tuning of model parameters requires certain robustness from the system. And as the size of the model and the number of the parameters grows, computerized numerical tuning methods become indispensable.

Although the same tuning algorithm can be applied to model and control parameter tuning there exists some differences. If model parameters are being tuned, the reference signals for simulated outputs come straight from the chosen data period. Erroneous measurements and noise may cause some problems here. In control parameter tuning the desirable trajectories for controlled variables (if the actual reference signal is considered a too strict objective) are specified by the user which offers a greater degree of freedom.

Perfect reconstruction of a certain situation (i.e., repeating the correct events from a particular initial state in correct order with correct timing and nothing else) with a simulator is a challenging task in general. Therefore, the approximated input signals need to be considered as another source of error and uncertainty in the simulation results (along with the plant-model mismatch) which has to be kept in mind as the results are evaluated. This problem is not restricted to model parameter tuning only but is present as well within the simulation assisted controller tuning framework.

An age old (and still widely used) saying goes that computers are getting much faster in the future but the *status quo* still seems to be that the simulations are always run overnight. This probably results from the fact that the applied models become more and more complex requiring more computational capacity. Techniques like parallel processing will probably decrease notably the computation times in future. In the meantime, however, the computational efficiency and applicability of algorithms to different problems still needs to be considered. In this example case, one simulation run involved a settling period of 95 minutes after each parameter change and after that a simulation of 15 minutes for the quality measure evaluation. (Note that over 85 per cent of the time was spent on settling simulations due to the slow dynamics of the process.) The model was run about 40 times faster than real time which means it took about 7 days to complete the about 4000 simulations of the 27 global iteration steps with an ordinary office computer. Clearly, practical application of IRT to systems with this slow dynamics starts to become questionable. On the other hand, if an automated tuning environment is

available, it is not a huge sacrifice to spend half an hour to configure the tuning targets and see how the tuning is progressing after a day or two.

Using a well designed tuning environment would facilitate the modeling task in general in many ways. It would shorten the time spent on setting up the tuning case and reduce the number of mistakes. Automatic coordination of the simulation runs would reduce the unwanted variation on the quality measure values and speed up the overall tuning process since the length of the settling simulations could be determined more practically. A version management system of old model versions and automatic storing of tuning results would make it possible to continue tuning after unexpected problems without huge manual efforts. The tuning environment could also offer different tools for visualizing the data which helps the monitoring of results.

6 CONCLUSIONS

In the traditional CPA framework detecting the badly performing controllers is the first step in the performance improvement process. After that further analysis is required to discover the underlying reason for the bad performance – is it caused by process equipment malfunction, external disturbance or bad controller tuning? Only after that incorrect control parameter values are corrected by redesigning the controller. When the IRT method is applied, the amount of required diagnosis is diminished since the potential for performance improvement by controller tuning is automatically revealed and exploited by the local model between quality measures and parameters and the following parameter update step. If performance targets of a controller cannot be met it is an indication of a fault in the system that cannot be fixed by parameter tuning but needs more attention.

Earlier domain area expertise was a necessity for determining the key controllers from a large industrial process whose operation had the greatest effect on the overall system performance. When IRT is applied it is enough to determine a set of controllers that contain these key controllers along with other controllers of smaller importance and the experts can concentrate on how the overall targets of performance are expressed. As the decision space is now augmented with the control parameters of secondary importance, better solutions in general can be obtained since the interdependencies between individual controllers are taken into account.

A similar effect appears on model parameter tuning as well. Since efficient MVR methods are applied there is no more need to manually reduce the number of decision variables in order to make the estimation easier or numerically more reliable. If only the completely unknown parameters are estimated from data and parameters with some physical interpretation are fixed based on practical experience, an optimal solution cannot be achieved. Without underrating engineering knowhow, it is, however, more beneficial to use such estimated parameter values as initial values and include these parameters in optimization in order to correct possible approximation errors.

As mentioned above, the role of expertise changes when IRT is applied instead of traditional working practices. New challenges arise when the overall system performance needs to be determined explicitly in mathematical terms. This can be seen also as a possibility for the experts to improve their intuitive understanding of the system. If the results of parameter optimization are not satisfactory in some sense one has to revalue ones mental model of the system based on which the targets were set and consider changing the formulation of targets. By this way the iteration concerning targets of a

single low-level SISO control loop has been replaced by the iteration on a higher abstraction level where the performance of the overall system is defined.

Despite the current tendency to design large control concepts for larger and more complex process units, the basic design process still relies on the traditional reductionist engineering paradigm in which the analysis of large systems begins with a partition into smaller subsystems. If one wants to study and improve plant level performance, emergent higher-level quality measures are needed to describe the performance targets and efficient statistical multivariate methods are needed to capture the dependencies between the targets and the system parameters. Bars *et al.* conclude that the design of very large control systems still presents a challenge to control theory and new theories are needed to handle complex systems involving a large number of control loops [4]. The IRT method can be seen as an attempt to clear the road for the large scale design methods being a manifestation of a paradigm change in engineering. It can be applied in many different settings for control, process and model parameter tuning as described in Chapter 4.

In Chapter 3 the underlying optimization problem was characterized and the selection of appropriate mathematical methods for solving it was discussed. It turns out that fairly simple methods can be applied if only the correct level of analysis is selected. Simplicity of analysis becomes a necessity since rigorous bottom-up analyzing methods apparently are not scalable to high dimensional problems. Complex dynamic underlying phenomena can be examined successfully on the higher abstraction level using static linear models. MVR methods like PLS handle effectively noise and collinearities in data and capture relevant phenomena even from relatively scarce sets of high dimensional data. The model-based approach of IRT has also other advantages than the reduction of noise effects. Since the local linear model compresses the underlying dependencies of system parameters and quality measures, it can be applied in a more profound analysis to detect, for instance, conflicting or parallel targets.

Application of IRT often benefits from guidance given by the domain area expert even though it does not in general necessitate interactive participation from the user during the tuning procedure. The role of human decision maker is emphasized at the end of the tuning procedure. In practice the decision makers need different visualization tools for assistance in order to get a grip of the situation. Watching solely the cost function and quality measure trends does not reveal the underlying causalities and can be in some cases even misleading since the values of the quality measures tend to approach the optimum with non-uniform rates.

IRT has been designed primarily for optimizing the numerical parameters of large scale control structures and dynamic process models. Its components have become chosen because of their good applicability to the problem. However, some weaknesses naturally still remain with the IRT method and when it is applied inappropriately these minor flaws may turn into major problems and hinder one from achieving satisfactory results.

First of all, being basically a heuristic LO approach any theoretical guarantees of convergence to global optimum cannot be given for IRT since the shape of the overall cost function is unknown and observations are noise corrupted. However, if the problem can be formulated decently and good initial values are available, some improvement on system performance can be achieved if that is only possible via parameter tuning. The final solution is searched for using decision making among conflicting targets which

affects the location of the global optimum making it furthermore a less meaningful concept.

Another challenging issue is comprehensive excitation of the system with respect to both parameter variation and input signal generation. If the variation of some parameters is of totally wrong magnitude either numerical problems in simulation or no effect on the simulation results are obtained, corresponding to too large and too small parameter variation. Finding a suitable level for parameter variation requires domain area expertise. Determining input signals for tuning simulations belongs to the task of target determination and is even more challenging. The applied simulation sequence should encompass a sufficiently representative set of events (production in different operating points and under different disturbances, etc.) that describes the operation of the system well. Naturally, one has to find a compromise between the representativeness and practicality of the tuning procedure.

By far the most time-consuming part of the tuning procedure is the time signal series generation, no matter whether they are computed on a simulation model or recorded from the actual process. In that sense minimization of the function evaluations during the tuning procedure is a worthwhile goal for further development of the IRT method. In practice this would mean a more reliable and consistent update direction and step length computation. For example, the use of second order derivative information in the vicinity of the optimum could improve the convergence speed.

7 REFERENCES

1. Anon. (2008), *APROS - The Advanced Process Simulation Environment*, VTT (Technical Research Centre of Finland), (referred 14.2.2009), http://apros.vtt.fi/.
2. Åström, K.J., Hägglund, T. (2001), *The future of PID control*, Control Engineering Practice, 9(11), pp. 1163-1175.
3. Åström, K.J., Wittenmark, B. (1995), *Adaptive control*, 2. edition, Addison-Wesley, Reading MA, USA, 574 p.
4. Bars, R., Colaneri, P., de Souza, C.E., Allgöwer, F., Kleimenov, A., Scherer, C. (2006), *Theory, Algorithms and Technology in the Design of Control Systems,* Annual Reviews in Control, 30(1), pp. 19-30.
5. Bristow, D.A. Tharayil, M. Alleyne, A.G. (2006), *A survey of iterative learning control*, IEEE Control Systems Magazine, 26(3), pp. 96-114.
6. Campi, M.C., Lecchini, A., Savaresi, S.M. (2000), *Virtual Reference Feedback Tuning (VRFT): a new direct approach to the design of feedback controllers*, Proceedings of the 39th IEEE Conference on Decision and Control. Sydney, Australia, pp. 623-629.
7. Cox, R.K., Smith, J.F., Dimitratos, Y. (2006), *Can simulation technology enable a paradigm shift in process control? Modeling for rest of us*, Computers and Chemical Engineering, 30(10-12), pp. 1542-1552.
8. Dayal, B.S., MacGregor, J.F. (1997), *Improved PLS Algorithms*, Journal of Chemometrics, 11(1), pp. 73-85.
9. Dochain, D., Marquardt, W., Won, S.C., Malik, O., Kinnaert, M. (2006), *Monitoring and Control of Process and Power Systems: Towards New Paradigms*, Annual Reviews in Control, 30(1), pp. 69-79.
10. Gill, P.E., Murray, W., Saunders, M.A. (2002), *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization*, SIAM Journal on Optimization, 12(4), pp. 979-1006.
11. Glad, T., Ljung, L. (2000), *Control Theory – Multivariable and Nonlinear Methods*, Taylor & Francis, Bodmin, Great Britain, 467 p.
12. Gullichsen, J., Fogelholm, C.-J. (2000), *Papermaking Science and Technology 6, Chemical pulping*, Fapet Oy, Helsinki, Finland, 1190p.
13. Gustafson, R.R., Sleicher, C.A., McKean, W.T., Finlayson, B.A. (1983), *Theoretical Model of the Kraft Pulping Process*, Ind. Eng. Chem. Process Des. Dev., 22, pp. 87-96.

14. Grimble, M.J., Uduehi, D. (2001), *Process Control Loop Benchmarking and Revenue Optimization*, Proceedings of the 2001 American Control Conference, June 25-27, Arlington, VA, USA, pp. 4313-4327.

15. Hägglund, T. (1995), *A control-loop performance monitor*, Control Engineering Practice, 3(11), pp. 1543-1551.

16. Hägglund, T. (1999), *Automatic detection of sluggish control loops*, Control Engineering Practice, 7(12), pp. 1505-1511.

17. Halmevaara, K., Hyötyniemi, H. (2004), *Process Performance Optimization Using Iterative Regression Tuning*, Report 139, Control Engineering Laboratory, Helsinki University of Technology, Helsinki, Finland, 79 p.

18. Halmevaara, H. Hyötyniemi, H. (2004), *Iterative Simulation Based Multivariate Control Parameter Tuning Method*, Proceedings of the 5th EUROSIM Congress, September 6-10, Paris, France.

19. Halmevaara, K., Hyötyniemi, H. (2005), *Performance Optimization of Large Control Systems – Case Study on a Continuous Pulp Digester*, Proceedings of the 16th IFAC World Congress, July 3-8, Prague, Czech Republic.

20. Halmevaara, K., Hyötyniemi, H. (2006), *Data-based Parameter Optimization of Dynamic Simulation Models*, Proceedings of the 47th Conference on Simulation and Modelling (SIMS 2006), September 27-29, Helsinki, Finland, pp. 69-73.

21. Halmevaara, K., Hyötyniemi, H. (2006), *Application of Elastic Intuitions to Process Engineering*, Proceedings of the 9th Scandinavian Conference on Artificial Intelligence (SCAI 2006), October 25-27, Espoo, Finland.

22. Halmevaara, K., Hyötyniemi, H. (2007), *Dynaamisten simulointimallien parametrien virittäminen datapohjaisilla tilastollisilla menetelmillä*, Proceedings of Automaatio07, March 27-28, Helsinki, Finland.

23. Halmevaara, K., Hyötyniemi, H. (2007), *Tuning of multi-parameter systems using multivariate regression and numerical optimization methods*, Proceedings of the 6th International Conference on Intelligent Processing and Manufacturing of Materials (IPMM 2007), June 25-29, Salerno, Italy.

24. Hang, C.C., Åström, K.J., Wang, Q.C. (2002), *Relay feedback auto-tuning of process controllers a tutorial review*, Journal of Process Control, 12(1), pp. 143-162.

25. Hang, C.C., Sin, K.K. (1991), *On-Line Auto Tuning of PID Controllers Based on the Cross-Correlation Technique*, IEEE Transactions on Industrial Electronics, 38(6), pp. 428-437.

26. Harris, T.J. (1989), *Assessment of control loop performance*, Canadian Journal of Chemical Engineering, 67(5), pp. 856-861.

27. Harris, T.J., Boudreau, F., MacGregor, J.F. (1996), *Performance Assessment of Multivariable Feedback Controllers*, Automatica, 32(11), pp. 1505-1518.

28. Harris, T.J., Seppala, C.T., Desborough, L.D. (1999), *A review of performance monitoring and assessment techniques for univariate and multivariate control systems*, Journal of Process Control, 9(1), pp. 1-17.

29. Harris, T.J., Seppala, C.T., Jofriet, P.J., Surgenor, B.W. (1996), *Plant-wide feedback control performance assessment using an expert-system framework*, Control Engineering Practice, 4(9), pp. 1297-1303.

30. He, K., Dong, S., Zheng, L. (2006), *Service-Oriented Grid Computation for Large-Scale Parameter Estimation in Complex Environmental Modeling*, Proceedings of the ACM symposium on Applied computing, Dijon, France, pp. 741-745.

31. Hjalmarsson, H., Gunnarsson, S., Gevers, M. (1994), *A Convergent Iterative Restricted Complexity Control Design Scheme*, Proceedings of the 33rd IEEE Conference on Decision and Control, Orlando, FL, p. 1735-1740.

32. Huang, B., Shah, S.L. (1999), *Performance assessment of control loops*, Springer Verlag, London, 255 p.

33. Huang, B., Shah, S.L., Miller, R. (2000), *Feedforward Plus Feedback Controller Performance Assessment of MIMO Systems*, IEEE Transactions on Control Systems Technology, 8(3), pp. 580-587.

34. Hyötyniemi, H. (2001), *Multivariate Regression – Techniques and Tools*, Report 125, Control Engineering Laboratory, Helsinki University of Technology, Helsinki, Finland, 207 p.

35. Hyötyniemi, H. (2003), *Emergence and Complex Systems – Towards New Practices for Industrial Automation?*, In: Meech, J.A., Kawazoe, Y., Kumar V., Maguire, J.F. (2005), Intelligence in a Small Materials World: Selected Papers from IPMM-2003, the Fourth International Conference on Intelligent Processing and Manufacturing of Materials, DEStech Publications, PA, USA, pp. 28-60.

36. Hyötyniemi, H. (2006), *Neocybernetics in Biological Systems*, Report 151, Control Engineering Laboratory, Helsinki University of Technology, Helsinki, Finland, 273 p.

37. Jämsä-Jounela, S-L., Poikonen, R., Halmevaara, K. (2002), *Evaluation of Level Control Performance*, Proceedings of the 15th IFAC World Congress, July 21-26, Barcelona, Spain.

38. Jämsä-Jounela, S-L., Poikonen, R., Vatanski, N., Rantala, A. (2003), *Evaluation of control performance: methods, monitoring tool and applications in a flotation plant*, Minerals Engineering, 16(11), pp. 1069-1074.

39. Jelali, M. (2006), *An Overview of Control Performance Assessment Technology and Industrial Applications*, Control Engineering Practice, 14(5), pp. 441-466.

40. Jun, M., Safonov, M.G. (1999), *Automatic PID Tuning: An Application of Unfalsified Control*, Proceedings of the IEEE International Symposium on Computer Aided Control System Design (CACSD), August 22-27, Hawai, USA, pp. 328-333.

41. Juslin, K. (2005), *A Companion Model Approach to Modelling and Simulation of Industrial Processes*, Doctoral Thesis, Helsinki University of Technology, 155 p.

42. Karhela, T. (2002), *A Software Architecture for Configuration and Usage of Process Simulation Models – Software Component Technology and XML-based Approach,* Technical Research Centre of Finland, VTT Publications 479, Espoo, Finland, 129 p.

43. Karhela, T. (2005), *Prosessilaitoksen elinkaarenaikaisen tiedonhallinnan palvelukehys*, Proceedings of Automaatio05, September 6-9, Helsinki, Finland.

44. Kettunen, A., Paljakka, M. (2006), *Process Simulation in Power Plant Design*, Proceedings of the 47th Conference on Simulation and Modelling (SIMS 2006), September 27-29, Helsinki, Finland, pp. 176-181.

45. Killingsworth, N.J., Krstić, M. (2006), *PID Tuning Using Extremum Seeking*, IEEE Control Systems Magazine, 26(1), pp. 70-79.

46. Klose, T., Kunze, D., Sandner, T., Schenk, H., Lakner, H., Schneider, A., Schneider, P. (2005), *Stress Optimization of a Micromechanical Torsional Spring*, Proceedings of the NSTI Nanotechnology Conference and Trade Show (Nanotech 2005), May 8-12, Anaheim, USA.

47. Konda, N.V.S.N.M., Rangaiah, G.P. (2007), *Performance Assessment of Plantwide Control Systems of Industrial Processes*, Industrial & engineering chemistry research, 46(4), pp. 1220-1231.

48. Kondelin, K., Karhela, T., Laakso. P. (2004), *Service Framework Specification for Process Plant Lifecycle*, VTT Research Notes, 2277, Espoo, Finland, 123 p.

49. Laakso, P., Paljakka, M., Kangas, P., Helminen, A., Peltoniemi, J., Ollikainen, T. (2005), *Methods of simulation-assisted automation testing*, VTT Research Notes, 2289, Espoo, Finland, 59 p.

50. Lequin, O., Gevers, M., Mossberg, M., Bosmans, E., Triest, L. (2003), *Iterative feedback tuning of PID parameters: comparison with classical tuning rules*, Control Engineering Practice, 11(9), pp. 1023-1033.

51. Lewis, P.H., Yang, C. (1997), *Basic Control Systems Engineering*, Prentice-Hall, New Jersey, USA, 450 p.

52. Liu, Y.B., Batelaan, O., De Smedt, F., Poórová, J., Velcická, L. (2005), *Automated calibration applied to a GIS-based flood simulation model using PEST*, In: van Alphen, J., van Beek, E., Taal, M. (2005), Floods, from defense to management, Taylor-Francis, London, UK, pp. 317–326.

53. Ljung, L. (1999), *System Identification: Theory for the User*, Prentice-Hall, New Jersey, USA, 672 p.

54. Longman, R.W. (2000), *Iterative learning control and repetitive control for engineering practice*, International Journal of Control, 73(10), pp. 930-954.

55. Lopes, J.A., Costa, P.F., Alves, T.P., Menezes, J.C. (2004), *Chemometrics in Bioprocess Engineering: Process Analytical Technology (PAT) Applications*, Chemometrics and Intelligent Laboratory Systems, 74(2), pp. 269-275.

56. Luyben, W.L. (1990), *Process Modeling, Simulation, and Control for Chemical Engineers*, McGraw-Hill Publishing, Singapore, 725 p.

57. Miller, R.E. (2000), *Optimization – Foundations and Applications*, John Wiley & Sons, New York, NY, USA, 653 p.

58. Miettinen, K. (1999), N*onlinear Multiobjective Optimization*, Kluwer Academic Publishers, Norwell MA, 298 p.

59. Miettinen, K., (2001), *Some Methods for Nonlinear Multi-objective Optimization*, Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), March 7-9, Zurich, Switzerland, pp. 1-20.

60. Miettinen K., Mäkelä M.M. (2006), *Synchronous Approach in Interactive Multiobjective Optimization*, European Journal of Operational Research, 170(3), pp. 909-922.

61. Nelles, O. (2001), *Nonlinear System Identification*, Springer-Verlag, Germany, 785 p.

62. O'Dwyer, A. (2003), *Handbook of PI and PID Controller Tuning Rules*, Imperial College Press, Singapore, 375 p.

63. Paulonis, M.A., Cox, J.W. (2003), *A practical approach for large-scale controller performance assessment, diagnosis, and improvement*, Journal of Process Control, 13(2), pp. 155-168.

64. Pesonen, L.T.T., Salminen, S.J., Ylén, J-P., Riihimäki, P. (2008), *Dynamic Simulation of Product Process*, Simulation Modelling Practice and Theory, 16(8), pp. 1091-1102.

65. Pindyck, R.S., Rubinfeld, D.L. (1991), *Econometric Models and Economic Forecasts*, Third Edition, McGraw & Hill, New York, USA, 596 p.

66. Pintér, J.D. (2007), *Global Optimization – Models, Algorithms, Software, and Algorithms*, Course material of Global Optimization seminar at Helsinki School of Economics, 19.-20.3.2007.

67. Poeter, E.P., Hill, M.C. (1999), *UCODE, a computer code for universal inverse modeling*, Computers & Geosciences, 25(4), pp. 457-462.

68. Robbins, H., Munro, S. (1951), *A Stochastic Approximation Method*, The Annals of Mathematical Statistics, 22(3), pp. 400-407.

69. Saeki, M. (2003), *Unfalsified Control Approach to Parameter Space Design of PID controllers*, Proceedings of the 42nd IEEE Conference on Decision and Control (CDC2003), December 9-12, Maui, HI, USA, pp. 786-791.

70. Safonov, M.G., Tsao, T-C. (1994), *The Unfalsified Control Concept and Learning*, Proceedings of the 33rd IEEE Conference on Decision and Control, Lake Buena Vista, FL, USA, pp. 2819-2824.

71. Smith, C.C., Williams, T.J. (1974), Mathematical Modeling, Simulation and Control of the Operation of Kamyr Continuous Digester for Kraft Process, In: Wisnewski, P.A., Doyle, F.J. III, Kayihan, F. (1997), Fundamental Continuous-Pulp-Digester Model for Simulation and Control, AIChE Journal, 43(12), pp. 3175-3192.

72. Schneider, P., Schneider, A., Schwarz, P. (2002), *A modular approach for simulation-based optimization of MEMS*, Microelectronics Journal, 33(1-2), pp. 29-38.

73. Spall, J.C. (2003), *Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control*, John Wiley & Sons, New York, USA, 618p.

74. Stanfelj, N., Marlin, T.E., MacGregor, J.F. (1993), *Monitoring and diagnosing process control performance: The single-loop case*, Industrial & Engineering Chemistry Research, 32(2), pp. 301-314.

75. Tahvonen, T. (2006), *Methods and Tools for Simulation Assisted Process Automation Testing*, Master's thesis, Helsinki University of Technology, 73 p.

76. Thode, H.C.Jr. (2002), *Testing for normality*, Marcel Dekker, New York, USA, 479 p.

77. Thornhill, N.F., Hägglund, T. (1997), *Detection and diagnosis of oscillation in control loops*, Control Engineering Practice, 5(10), pp. 1343-1354.

78. Vanderplaats, G.N. (1984), *Numerical Optimization Techniques for Engineering Design*, McGraw-Hill, NY, USA, 333 p.

79. Wold, S., Sjöström, M., Eriksson, L. (2001), *PLS-regression: a basic tool for chemometrics*, Chemometrics and Intelligent Laboratory Systems, 58(2), pp. 109-130.

80. Xu J.-X., D. Huang, S. Pindi (2008), *Optimal Tuning of PID Parameters Using Iterative Learning Approach*, SICE Journal of Control, Measurement, and System Integration 1(2), pp. 143-154.

81. Ylén, J-P., Paljakka, M., Karhela, T., Savolainen, J., Juslin, K. (2005), *Experinences on Utilising Plant Scale Dynamic Simulation in Process Industry*, Proceedings of the 19th European Conference on Modelling and Simulation ECMS 2005, pp. 685-690.

82. Yu, L.X., Lionberger, R.A., Raw, A.S., D'Costa, R., Wu, H., Hussain, A.S. (2004), *Applications of Process Analytical Technology to Crystallization Processes*, Advanced Drug Delivery Reviews, 56(3), pp. 349-369.

83. Zabinsky, Z.B. (2003), *Stochastic Adaptive Search for Global Optimization*, Kluwer Academic Publishers, 224 p.

APPENDIX I: APROS

The following short introduction of Apros (Advanced Process Simulation) software is based on the material presented on the official Apros web pages [1]. Apros is a dynamical modeling and simulation environment designed for professional usage on the different fields of process industry. Originally Apros, developed by the Technical Research Centre of Finland (VTT) and Fortum Plc (former Imatran Voima) in the 1980s, was intended for modeling nuclear and conventional power plants. Nowadays, there exist several different releases for different applications, for example, Apros Combustion, Apros Nuclear, and Apros Paper. Application examples are presented in [81].

Apros provides the user with extensive model component libraries that enable efficient construction of large plant-wide process models in reasonable time. Libraries contain numbers of different component prototypes for basic process elements, such as pipes and tanks, and for more complex devices like heat exchangers, valves, pumps, etc. Also automation and electrical components have their own libraries. The library prototypes can be adjusted to correspond to the actual hardware in use by changing their parameterizations. Model components can be connected into large networks using different connection types for fluid flows and information signals. When the components are attached to the model, the underlying mathematical model equations are created automatically making the modeling process effortless.

The user can select between several accuracy levels, i.e., thermo hydraulic modeling principles, in Apros. In the simplest models, the same temperatures and velocities are assumed for gaseous and liquid flows and the model is based on the mass, momentum and energy conservation equations only. In the more rigorous models, however, mass, momentum and energy equations are solved separately for liquid and gaseous phases. These models involve also calculation of friction and heat conduction and transfer. Apros applies different solvers for different model accuracy levels. During simulation model variables are solved using a grid discretization approach in which the values of state variables (e.g., pressure and enthalpy) are solved in the *nodes* of the mesh, and flow related variables are computed in the *branches* between the nodes. Water and steam properties in nodes are obtained from accurate look-up tables. Alongside the thermal hydraulic network, Apros involves a composition network for material properties and concentrations of different components of fluids. It is possible to model chemical reactions between different substances as well.

Different methods of numerical integration can be applied to solve the set of partial differential equations (PDE) constituting the process model, for example, the implicit Euler method and Trapezoid method. Apros selects an appropriate method automatically without user intervention. Implicit time integration results in a set of algebraic equations

that need to be (linearized and) solved numerically on each time step. Juslin [41] presents a detailed introduction to the solution principles applied in Apros models.

Apros utilizes a graphical user interface called Grades (Figure 41), or it can be used with a lower lever command line tool. Apros commands can be assembled into command queue files that determine the events during a simulation run. Apros simulation engine can read and write time series data from/to text files. Like this the measurement data from the actual process plant can be used as a time-dependent boundary condition in simulation or the simulation results can be stored for later analysis. Apros models are relatively easy to connect with external model components, for example, separate models of the automation system, using the OPC interface.

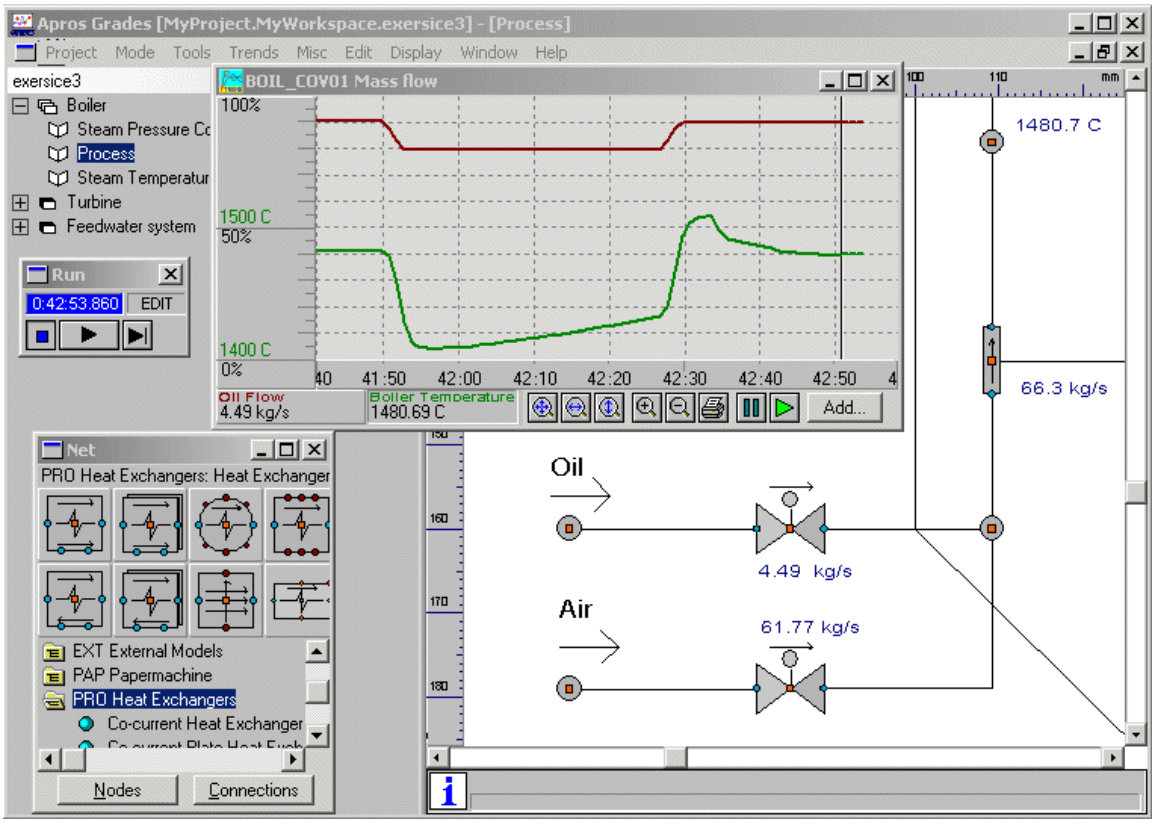

Figure 41. An example view of the Apros modeling environment using the Grades graphical user interface [1]. Process components are shown on the drawing canvas and simulation results on the trend windows. Model components, like different heat exchangers, can be browsed on the library toolbar.

HELSINKI UNIVERSITY OF TECHNOLOGY CONTROL ENGINEERING

Editor: H. Koivo

Report 149 Kantola, K.
Modelling, Estimation and Control of Electroless Nickel Plating Process of Printed Circuit Board Manufacturing. March 2006.

Report 150 Virtanen, T.
Fault Diagnostics and Vibration Control of Paper Winders. June 2006.

Report 151 Hyötyniemi, H.
Neocybernetics in Biological Systems. August 2006.

Report 152 Hasu, V.
Radio Resource Management in Wireless Communication: Beamforming, Transmission Power Control, and Rate Allocation. June 2007.

Report 153 Hrbček, J.
Active Control of Rotor Vibration by Model Predictive Control - A simulation study. May 2007.

Report 154 Mohamed, F. A.
Microgrid Modelling and Online Management. January 2008.

Report 155 Eriksson, L., Elmusrati, M., Pohjola, M. (eds.)
Introduction to Wireless Automation - Collected papers of the spring 2007 postgraduate seminar. April 2008.

Report 156 Korkiakoski, V.
Improving the Performance of Adaptive Optics Systems with Optimized Control Methods. April 2008.

Report 157 Al.Towati, A.
Dynamic Analysis and QFT-Based Robust Control Design of Switched-Mode Power Converters. September 2008.

Report 158 Eriksson, L.
PID Controller Design and Tuning in Networked Control Systems. October 2008.

Report 159 Pohjoranta, A.
Modelling Surfactant Mass Balance with the ALE Method on Deforming 2D Surfaces. May 2009.

Report 160 Kaartinen, J.
Machine Vision in Measurement and Control of Mineral Concentration Process. June 2009.

Report 161 Hölttä, V.
Plant Performance Evaluation in Complex Industrial Applications. September 2009.

Report 162 Halmevaara, K.
Simulation Assisted Performance Optimization of Large-Scale Multiparameter Technical Systems. September 2009.

ISBN 978-952-248-098-9

ISSN 0356-0872

Yliopistopaino, Helsinki 2009