Piotr Tatjewski

AIC

Advances in Industrial Control

# Advanced Control of Industrial Processes

## Structures and Algorithms

Springer

# Advances in Industrial Control

## *Other titles published in this Series:*

*Digital Controller Implementation and Fragility*
Robert S.H. Istepanian and James F. Whidborne (Eds.)

*Optimisation of Industrial Processes at Supervisory Level*
Doris Sáez, Aldo Cipriano and Andrzej W. Ordys

*Robust Control of Diesel Ship Propulsion*
Nikolaos Xiros

*Hydraulic Servo-systems*
Mohieddine Jelali and Andreas Kroll

*Strategies for Feedback Linearisation*
Freddy Garces, Victor M. Becerra, Chandrasekhar Kambhampati and Kevin Warwick

*Robust Autonomous Guidance*
Alberto Isidori, Lorenzo Marconi and Andrea Serrani

*Dynamic Modelling of Gas Turbines*
Gennady G. Kulikov and Haydn A. Thompson (Eds.)

*Control of Fuel Cell Power Systems*
Jay T. Pukrushpan, Anna G. Stefanopoulou and Huei Peng

*Fuzzy Logic, Identification and Predictive Control*
Jairo Espinosa, Joos Vandewalle and Vincent Wertz

*Optimal Real-time Control of Sewer Networks*
Magdalene Marinaki and Markos Papageorgiou

*Process Modelling for Control*
Benoît Codrons

*Computational Intelligence in Time Series Forecasting*
Ajoy K. Palit and Dobrivoje Popovic

*Modelling and Control of mini-Flying Machines*
Pedro Castillo, Rogelio Lozano and Alejandro Dzul

*Rudder and Fin Ship Roll Stabilization*
Tristan Perez

*Hard Disk Drive Servo Systems* (2nd Ed.)
Ben M. Chen, Tong H. Lee, Kemao Peng and Venkatakrishnan Venkataramanan

*Measurement, Control, and Communication Using IEEE 1588*
John Eidson

*Piezoelectric Transducers for Vibration Control and Damping*
S.O. Reza Moheimani and Andrew J. Fleming

*Manufacturing Systems Control Design*
Stjepan Bogdan, Frank L. Lewis, Zdenko Kovačić and José Mireles Jr.

*Windup in Control*
Peter Hippe

*Nonlinear $H_2/H_\infty$ Constrained Feedback Control*
Murad Abu-Khalaf, Jie Huang and Frank L. Lewis

*Practical Grey-box Process Identification*
Torsten Bohlin

*Modern Supervisory and Optimal Control*
Sandor Markon, Hajime Kita, Hiroshi Kise and Thomas Bartz-Beielstein

*Wind Turbine Control Systems*
Fernando D. Bianchi, Hernán De Battista and Ricardo J. Mantz

*Advanced Fuzzy Logic Technologies in Industrial Applications*
Ying Bai, Hanqi Zhuang and Dali Wang (Eds.)

*Practical PID Control*
Antonio Visioli

*Soft Sensors for Monitoring and Control of Industrial Processes*
Luigi Fortuna, Salvatore Graziani, Alessandro Rizzo and Maria Gabriella Xibilia

Piotr Tatjewski

# Advanced Control of Industrial Processes

## Structures and Algorithms

Springer

Piotr Tatjewski, PhD, DSc, Professor of Technical Sciences
Warsaw University of Technology
Institute of Control and Computation Engineering
Nowowiejska 15/19
00-665 Warszawa
Poland

British Library Cataloguing in Publication Data
Tatjewski, Piotr
Advanced control of industrial processes : structures and
algorithms. - (Advances in industrial control)
1. Predictive control - Mathematics 2. Fuzzy algorithms
I. Title
629.8'015181
ISBN-13: 9781846286346
ISBN-10: 1846286344

Library of Congress Control Number: 2006936016

Advances in Industrial Control series ISSN 1430-9491
ISBN 978-1-84628-634-6 e-ISBN 1-84628-635-2 Printed on acid-free paper



9 8 7 6 5 4 3 2 1

Springer Science+Business Media
springer.com

# Advances in Industrial Control

Professor Emeritus O.P. Malik
Department of Electrical and Computer Engineering
University of Calgary
2500, University Drive, NW
Calgary
Alberta
T2N 1N4
Canada

Professor K.-F. Man
Electronic Engineering Department
City University of Hong Kong
Tat Chee Avenue
Kowloon
Hong Kong

Professor G. Olsson
Department of Industrial Electrical Engineering and Automation
Lund Institute of Technology
Box 118
S-221 00 Lund
Sweden

Professor A. Ray
Pennsylvania State University
Department of Mechanical Engineering
0329 Reber Building
University Park
PA 16802
USA

Professor D.E. Seborg
Chemical Engineering
3335 Engineering II
University of California Santa Barbara
Santa Barbara
CA 93106
USA

Doctor K.K. Tan
Department of Electrical Engineering
National University of Singapore
4 Engineering Drive 3
Singapore 117576

Professor Ikuo Yamamoto
Kyushu University Graduate School
Marine Technology Research and Development Program
MARITEC, Headquarters, JAMSTEC
2-15 Natsushima Yokosuka
Kanagawa 237-0061
Japan

To the memory of my mother

# Series Editors' Foreword

The series *Advances in Industrial Control* aims to report and encourage technology transfer in control engineering. The rapid development of control technology has an impact on all areas of the control discipline. New theory, new controllers, actuators, sensors, new industrial processes, computer methods, new applications, new philosophies..., new challenges. Much of this development work resides in industrial reports, feasibility study papers and the reports of advanced collaborative projects. The series offers an opportunity for researchers to present an extended exposition of such new work in all aspects of industrial control for wider and rapid dissemination. Industrial process control usually requires process unit control and global plant-wide control. The question is how to model and solve these often large-scale control problems. In practice, industrial and control engineers find a way to make these processes work and then refine and optimize the control structures. The involvement of the academic control community in the solution of these problems is often third-hand and usually *after* the process is up and running. Consequently, the amount of teaching and research devoted to industrial process supervisory control tends to be rather small when compared with activity in control loop design. This is a pity because these higher-level problems in process control are both important and challenging.

The two seminal textbooks in this field are *Theory of Hierarchical Multilevel Systems* by M.D. Mesarovic, D. Macko and Y. Takahara (Academic Press, New York, 1970) and *Control and Coordination in Hierarchical Systems* by W. Findeisen, F.N. Bailey, M. Brdys, K. Malinowski, P. Tatjewski and A. Wozniak (J. Wiley and Sons, Chichester, U.K., 1980). These books provided the mental models and the system vocabulary to bring a concrete framework to the engineering community tackling large-scale industrial process control. The key concept in this ordering was the *hierarchy* and the key tool was *optimisation*. Since these seminal texts emerged, the use of a hierarchical structure in industrial control is usually implicitly or explicitly present. Many global plant control schemes are designed around a hierarchical layered framework because this enables objectives, methods and operation to be

clearly and unambiguously stated and implemented. The hierarchical structure often facilitates the easy transfer of knowledge of the plant control objectives and methods as new engineers join the operational team. Enhancing the effectiveness of global plant control becomes an easier task if the control is decomposed into layers that are usually time-frame decoupled, too.

Despite the passage of time, *optimisation* remains the key tool but what has changed are the optimisation methods. Optimisation in the 1970s usually meant static optimisation techniques whereas today constrained dynamic optimisation of complex nonlinear processes is routinely feasible. Model predictive control methods are versatile and commonly found in the higher reaches of the process control hierarchy, as well as providing the control designs in the lower control-loop and set-point-changeover levels.

Piotr Tatjewski was one of the original authorial team that produced the seminal text *Control and Coordination in Hierarchical Systems*. Consequently, it comes as no surprise to find that his *Advances in Industrial Control* monograph, *Advanced Control of Industrial Processes* has the sub-title *Structures and Algorithms*. For what the reader will find in this exemplary monograph is two chapters reviewing and extending the original concepts of the multi-layer hierarchical control structure and two chapters introducing in considerable scholarly depth the control algorithms of model fuzzy control and model predictive control. The two control algorithms can be considered for use at different levels and to achieve different objectives within the hierarchical control structure. Of course, the model predictive control tool facilitates optimisation whilst model fuzzy control can be viewed as a loop controller model devised to achieve good nonlinear process control performance in the direct control layer of the hierarchy.

The monograph has been written with considerable care given to the steps of review, exposition and demonstration. Industrially relevant examples have been chosen to demonstrate different aspects of the concepts under discussion. The careful presentation enables both the industrial engineer and the academic researcher to appreciate the context of the ideas being discussed before proceeding to the more challenging aspects of the exposition. Since sufficient information has been given about many of the various examples presented, enthusiastic readers may wish to repeat them for themselves.

Readers of all levels of attainment in industrial process control will find something of interest in this fine monograph. It is a very welcome new title in the *Advances in Industrial Control* series.

M.J. Grimble and M.A. Johnson
Glasgow, Scotland, U.K.

# Preface

The subject of this book is *advanced control* of industrial processes. Therefore, algorithms of advanced feedback control are mainly presented, but also on-line set-point optimization is discussed, in appropriate structures. A starting point for the topic defined in this way is a *multilayer control structure* of industrial processes. This structure enables reasonable and safe control and management through a decomposition and distribution of tasks and responsibilities into a few well-defined and simpler sub-tasks of a more homogeneous character, mutually interconnected. The basic layers of the multilayer structure are two feedback control layers (regulatory control layers) and the optimization layer.

Feedback control is now often carried out at two layers, especially for more complex, multivariable processes. The first and lowest one is the *direct control layer*, also called the basic control layer. Its main task is to maintain the process within desirable limits defined by the set-point values for its controllers (direct controllers). It is usually also equipped with certain control logic responsible for overriding the control designed for normal operating conditions, if a danger of violating constraints leading to an emergency state is encountered. Set-points of certain direct controllers of complex processes are now more and more often under supervisory control of advanced controllers, which can therefore be called the set-point controllers. They constitute a (higher) *set-point control layer*, which is even more commonly called a *constraint control layer*, as its controllers are usually responsible for controlling certain technological constraints influencing product quality.

As a result of the development of electronics and computer technology, the powerful DCS (Distributed Control System) and SCADA (Supervisory Control and Data Acquisition) systems are now a standard, enabling advanced realization of feedback control tasks. Thus, it is possible to apply, in real time, the advanced control techniques. They are usually understood as algorithms more complex than those based on the classic PID control law. These techniques are now applied mainly at the constraint control layer, where process nonlinearity, its multivariable nature and constraints play an important role. However, they can also be addressed to direct control loops which are

difficult for classic PID control, *e.g.*, due to large delays, signal constraints or nonlinearities.

The task of the *optimization layer* is to economically evaluate best set-point values for the feedback controllers. Due to an increased computing power it is now realistic to apply on-line optimization, that is optimal and automatic adjustment of the set-points to the varying external influences. However, frequent changes of the set-points calculated at the optimization layer require that the feedback controllers operate not only in the vicinity of one equilibrium point, but can cope with a wider range of variability of input and output process values as well. Real processes are generally nonlinear, therefore there is a need for nonlinear control systems. Moreover, the more precise the optimizer's model of the process and the more precise the stabilization of the optimal values by the feedback controllers, the higher the profits from the optimization. Therefore, the optimization usually sets new requirements for feedback control systems, creating a need for application of advanced control algorithms (in the sense defined earlier).

The subject of the book is situated within the scope of the discussed problems. In the first chapter general issues related to the control in the multilayer structure are discussed. Starting from basic control objectives, the question of a decomposition which leads to the structure is considered. Then the tasks, realization aspects and features of the direct control layer, the constraint control layer and the optimization layer are presented and discussed. Presentation of the multilayer control structure is supported by a carefully chosen worked example of a nonlinear chemical reactor, showing many aspects of the discussed topic, from a decomposition of the process dynamics to the set-point optimization.

The second chapter is devoted to nonlinear control algorithms using fuzzy structures of the Takagi-Sugeno (TS) type. After a short introduction to fuzzy logic and fuzzy nonlinear modeling, design procedures for discrete-time and continuous-time nonlinear TS fuzzy control algorithms are described, both for state-space and input-output process models. The questions of stability analysis of resulting nonlinear control systems are thoroughly addressed. It is not only the author's opinion that the TS fuzzy control is an efficient technique, relatively easy to design and convincing, since it can be treated as a natural nonlinear generalization of the classical linear control algorithms. In particular, it is shown to be a generalization of the well-established PID technique to the form of a nonlinear TS fuzzy PID controller. Constituting a systematic design alternative, the nonlinear fuzzy TS controllers are also a good solution in places where it is necessary to move from linear to nonlinear algorithms, without loosing the experience gathered.

Chapter 3 is devoted to predictive control algorithms, defined by a now commonly used acronym MPC (Model Predictive Control). The MPC is one of the advanced techniques which has achieved great, unquestionable success in practical applications, and has recently had a dominant impact on the direction of development of industrial control systems as well as scientific re-

search within the area of feedback control. There are several reasons for this. The MPC algorithms are indeed the first technique which directly takes into account constraints on both process inputs and outputs. It generates manipulated inputs while also considering internal interactions in the process, due to the direct use of the process model. Therefore, it is efficacious in application to multivariable processes, including those with a different number of manipulated inputs and controlled outputs. Moreover, the principles of operation and tuning of MPC algorithms are comprehensible, relatively easy to explain to engineering and operator staff – an important aspect when introducing new techniques to industrial practice. Needing more calculations at each sampling instant, initially the MPC algorithms were used mainly at the set-point control layer, where longer sampling periods are typical and the key questions are those of constraints and interactions. With increase in computing power and reliability and decrease in prices of processors it has become possible to apply the predictive algorithms also in direct control loops.

In Chapter 3 first linear predictive control algorithms are presented, concentrating on most important formulations: the DMC (Dynamic Matrix Control) algorithm, undoubtedly most popular in industrial process control practice and the GPC (Generalized Predictive Control) algorithm. Explicit versions (constraint free, leading to control laws) are discussed, as well as numerical ones, when at every sampling instant a numerical task of quadratic programming is solved on-line. Starting from linear formulations, the structures of basic nonlinear MPC algorithms are presented, paying particular attention to versions with linearizations, which are essential for effective applications. It is shown that these versions can be particularly easy to implement for nonlinear fuzzy models of the TS type. Problems of stability of MPC algorithms are also discussed, as well as questions of interpretation and adjustment of tuning knobs.

The fourth and last chapter is devoted to algorithms for set-point optimization. After a more general discussion about steady-state optimization in a multilayer control structure, steady-state optimization for control structures with MPC controllers is the subject of presentation. First, attention is devoted to a case when dynamics of disturbances can be comparable with the dynamics of the controlled process. In this case, the classical multilayer approach, with significantly different frequencies of intervention of different layers (the higher the layer, the slower the frequency) usually fails to result in a globally optimal control structure. However, applying an additional simplified steady-state optimization coordinated with the MPC dynamic optimization allows to improve the results. An interesting subject here is an algorithm integrating set-point and MPC dynamic optimizations. In the second part of the chapter, algorithms for on-line measurement-based iterative optimization (iterative improvement) of a steady-state operating point, under significant uncertainty caused by an imprecise model of the controlled process and/or errors in disturbance estimations, are presented. These algorithms are based mainly on the technique of integrated system optimization and parameter estimation.

In the book several important and well-established control structures and algorithms are presented. Starting from basic and known formulations (though supplemented with original views of the author, such as a new, alternative formulation of the GPC control law), the book also includes a series of research results obtained by the author, including those with his PhD students, concerning the nonlinear fuzzy control, the MPC algorithms and the set-point optimization techniques. Therefore, the book is addressed both to research staff and postgraduate students as well as to readers interested in the basic mechanisms of the presented techniques of advanced control, including engineers and practitioners. An appealing feature of the book is illustration of the presented concepts and algorithms by many worked examples in the text, as well as by results of many simulations based on industrial process models, stemming primarily from petrochemical and chemical industries.

This new book is based on a text published originally by the author in Polish in 2002, but several parts of this text have been improved or substantially changed when preparing this edition. In particular, certain topics have been deleted and new topics and research results have been included.

The author of this book is grateful to the colleagues and students from the Institute of Control and Computation Engineering, Warsaw University of Technology, for fruitful discussions, help and encouragement to write this book. In particular, the author is much indebted to Professor Władysław Findeisen, his esteemed teacher and to Professor Krzysztof Malinowski, longtime head of the University Priority Research Program in Control, Information Technology and Automation, supporting the author's research activities. The author is also very grateful to Professors P. D. Roberts from City University, London, and M. A. Brdyś from Birmingham University, for invitations to spend considerable time at these universities and for direct cooperation in research on steady-state optimizing control. The author is also thankful to his former PhD students and actual co-workers, in particular to Dr. Maciej Ławryńczuk and Dr. Piotr Marusak for cooperation in research on predictive control and for help in calculation of some examples and proofreading of the manuscript.

Acknowledgments are also due to the Polish Committee of Scientific Research and then the Polish Ministry of Scientific Research and Information Technology, for supporting the author's research from Polish budget funds in the form of research projects, in particular the one in the last two years, which contributed to this book.

Finally, the author is much indebted to his niece, Anna Basiukiewicz, a specialist in English language, who agreed to read and correct the final text of the book. Last but not least, the author owes a debt of gratitude to his wife Magda for her patience, understanding and support during the course of the book's preparation.

Warsaw, Poland, *Piotr Tatjewski*
August 2006

# Contents

# Notation

### General

| | |
|---|---|
| $x,\ y, ...$ | variables or constants, scalar or vector-valued |
| $n_x$ | dimensionality of vector $x$, $n_x = \dim x$ |
| $\mathbf{A}, \mathbf{B}, ...$ | real matrices |
| $x^T,\ \mathbf{A}^T$ | transpose of vector $x$, of matrix $\mathbf{A}$ |
| $\|x\|^2_{\mathbf{R}}$ | $x^T \mathbf{R} x$ |
| $\text{diag}\{a_1, ..., a_n\}$ | diagonal matrix with $a_1, ..., a_n$ on the diagonal |
| $(x, y)$ | ordered pair of elements $x$ and $y$, also vector $[x^T\ y^T]^T$ |
| $A(z^{-1})$ | algebraic polynomial in unit delay operator $z^{-1}$ |
| $E\{\cdot\}$ | expected value operator |
| $g(\cdot),\ f(\cdot), ...$ | scalar or vector functions |
| $g^{'}(x)$ | derivative of function $g$ at point $x$ |
| | for $g : \mathbb{R}^n \to \mathbb{R}$, $g^{'}(x) = [\frac{\partial g(x)}{\partial x_1}\ \cdots\ \frac{\partial g(x)}{\partial x_n}]$, |
| | for $g : \mathbb{R}^n \to \mathbb{R}^m$, $g^{'}(x) = \{\frac{\partial g_i(x)}{\partial x_j}\} = \begin{bmatrix} \frac{\partial g_1(x)}{\partial x_1} & \cdots & \frac{\partial g_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_m(x)}{\partial x_1} & \cdots & \frac{\partial g_m(x)}{\partial x_n} \end{bmatrix}$ |
| $\nabla g(x)$ | gradient, $\nabla g(x) = (g^{'}(x))^T$ |
| $g^{'}_x(x, y)$ | partial derivative of function $g$ with respect to $x$, at $(x, y)$ |

### Specific

| | |
|---|---|
| $\mu_C(\cdot)$ | membership function of the fuzzy set $C$ |
| $w^i(k)$ | activation level of $i$-th fuzzy inference rule at sample $k$ |
| $u(k)$ | manipulated variable (process control input) at sample $k$ |
| $y(k)$ | controlled variable (process controlled output) at sample $k$ |
| $x(k)$ | state of dynamic system at sample $k$ |

| | |
|---|---|
| $z(k)$ | measured disturbance at sample $k$ |
| $d(k)$ | unmeasured disturbance at process output at sample $k$ |
| $e(k)$ | control error at sample $k$, $e(k) = y^{sp}(k) - y(k)$ |
| $y^{sp}(k)$ | set-point for the controlled variable $y(k)$ at sample $k$ |
| $c$ | decision variable of the optimization layer, simultaneously: set-point for controllers of lower layers (Chapters 1 and 4) |
| $y(k+p\|k)$ | value of $y$ predicted for sample $k+p$ at current sample $k$ |
| $y^0(k+p\|k)$ | free component of $y(k+p\|k)$ |
| $\triangle y(k+p\|k)$ | forced component of $y(k+p\|k)$ |
| $s_j$ | $j$-th element of discrete unit step response |
| $D$ | dynamics horizon, *i.e.*, $s_j = \text{const.}$ for $j \geq D$ |
| $T_p$ | sampling period in a discrete dynamic system |
| $\tau$ | process time delay (defined as a number of sampling periods), excluding a unit discretization delay ($\tau \geq 0$) |
| $\bar{\tau}$ | overall time delay in a discrete-time model (defined as a number of sampling periods), including the discretization delay, $\bar{\tau} = \tau + 1$ |
| $F(\cdot)$ | model of steady-state input-output process mapping |
| $F_*(\cdot)$ | real (unknown) steady-state input-output process mapping |

*Parameters of model predictive controllers:*

| | |
|---|---|
| $N$ | prediction horizon (defined as a number of sampling periods) |
| $N_u$ | control horizon (defined as a number of sampling periods) |
| $N_1$ | initial time for summing control errors in the predictive controller cost function ($1 \leq N_1$, usually $N_1 = \tau + 1$) |
| $N_{cw1},\ N_{cw}$ | lower and upper bound of the constraint window (defined as numbers of sampling periods), $N_1 \leq N_{cw1} < N_{cw} \leq N$ |
| $\mathbf{\Psi}(p)$ | weighting matrix for control errors predicted for sample $k+p$ |
| $\underline{\mathbf{\Psi}}$ | $\underline{\mathbf{\Psi}} = \text{diag}\{\mathbf{\Psi}(N_1), ..., \mathbf{\Psi}(N)\}$ |
| $\mathbf{\Lambda}(p)$ | weighting matrix for control input moves for sample $k+p$ |
| $\underline{\mathbf{\Lambda}}$ | $\underline{\mathbf{\Lambda}} = \text{diag}\{\mathbf{\Lambda}(0), ..., \mathbf{\Lambda}(N_u - 1)\}$ |
| $\lambda$ | scalar weighting coefficient in the case when $\mathbf{\Lambda}(p) = \lambda \mathbf{I}$ |
| $\gamma$ | coefficient of first-order linear filter defining reference trajectory for controlled variables |

**Acronyms**

| | |
|---|---|
| ANFIS | Adaptive Neuro-Fuzzy Inference System |
| ARMAX | Auto-Regressive Moving Average with eXogenous input |
| ARX | Auto-Regressive with eXogenous input |
| DCS | Distributed Control System |

| | |
|---|---|
| ISOPE | Integrated System Optimization and Parameter Estimation |
| LMI | Linear Matrix Inequalities |
| LP | Linear Programming |
| MIMO | Multi-Input Multi-Output |
| PDC | Parallel Distributed Compensation |
| QP | Quadratic Programming |
| SCADA | Supervisory Control and Data Acquisition |
| SISO | Single-Input Single-Output |
| SQP | Sequential Quadratic Programming |

*Model predictive control algorithms:*

| | |
|---|---|
| CRHPC | Constrained Receding Horizon Predictive Control |
| DMC | Dynamic Matrix Control |
| FDMC | Fuzzy DMC |
| FGPC | Fuzzy GPC |
| FMPC | Fuzzy MPC |
| FMPCS | Fuzzy MPCS |
| GPC | Generalized Predictive Control |
| IDCOM | IDentification and COMmand |
| LSSO | Local Steady-State Optimization |
| MAC | Model Algorithmic Control |
| MPHC | Model Predictive Heuristic Control |
| MPC | Model Predictive Control (Model-based Predictive Control) |
| MPC-NO | MPC with Nonlinear Optimization |
| MPC-NPL | MPC with Nonlinear Prediction and Linearization |
| MPC-NPL+ | MPC-NPL algorithm with additional inner iteration loop |
| MPC-NSL | MPC Nonlinear with Successive Linearization |
| MPCS | MPC with State-space model |
| PFC | Predictive Functional Control |
| QDMC | Quadratic Dynamic Matrix Control |
| SMOC | Shell Multivariable Optimizing Controller |
| SSTO | Steady-State Target Optimization |

# 1

# Multilayer Control Structure[1]

## 1.1 Control System

Control problems exist and arise in numerous fields of human activity. Controlling a process (an object) can be defined as influencing it in such a way as to force it to operate in accordance with certain assumed requirements. This definition applies to all processes which undergo control. Thus, we can talk about control of technical objects, such as airplanes, electricity generators, technological processes in chemical reactors, distillation columns or sewage treatment plants, processes of transfer and exchange of information in telecommunications or computer networks, *etc.* We can also talk about the control of economic processes – in a company, in a holding or in an entire branch of economy (the word *management* is more commonly used here, instead of the word *control*), *etc.*

A controlled process is always surrounded by the environment in which it exists, undergoing controlled or uncontrolled influences of this environment. The controlled influences are generated by a control unit, *e.g.*, in a form of algorithms executed by an automatic control computer or in a form of decisions made by human beings. For example, an airplane is controlled by the pilot to enforce direction, height and other flight parameters. On the other hand, speed and direction of wind or air currents influence the flight parameters as well, but cannot be controlled. Similarly, a control computer or a human operator tries to achieve the desired parameters of technological processes in a chemical reactor or in a distillation column by enforcing appropriate values of selected process variables which influence its behavior (levels, flows, temperatures, *etc.*), counteracting changes in supply (raw materials, utilities) and in ambient conditions, which disturb a desired course of the process. Many more examples can be quoted, also within the range of economic or information

[1]Part of this chapter is a modified version of the text from Sections 1.1, 1.2 and 1.3 of the book Brdys, M.A. and Tatjewski, P., *Iterative Algorithms for Multilayer Optimizing Control*, copyright 2005 by Imperial College Press, used by permission.

technology processes. A significant common feature which we pay attention to in these examples is the fact that the process is not isolated from its environment, but rather that it undergoes external influences defined by certain input variables. The process input variables may be at the disposal of a control unit or may be not, thereby disturbing the behavior of the process from the point of view of the control unit. Therefore, the uncontrolled input variables are usually called *disturbances*. The process input variables, whose values can be changed by the control unit are usually called the process *manipulated variables* or the process *control inputs*.

Evaluation of the state of a controlled process, whether or not it fulfills the assumed requirements, whether or not the influence of manipulated inputs is correct, is done on the basis of measurements. More generally, it is done on the basis of observations of values and features of appropriate variables characterizing the process behavior. These variables are called *process output variables*. In a case of the control of a chemical reactor or a distillation column, examples of process outputs are parameters of a reacting or distilled mixture, such as temperature or composition, as well as parameters characterizing the state of technological apparatus (liquid levels, temperatures, pressures, *etc.*). Knowing objectives of control and analyzing values of the process outputs and those disturbances which are known (measured, estimated), the control unit makes decisions whether to maintain or appropriately change values of the control inputs. The general structure of a control system is presented in Fig. 1.1.

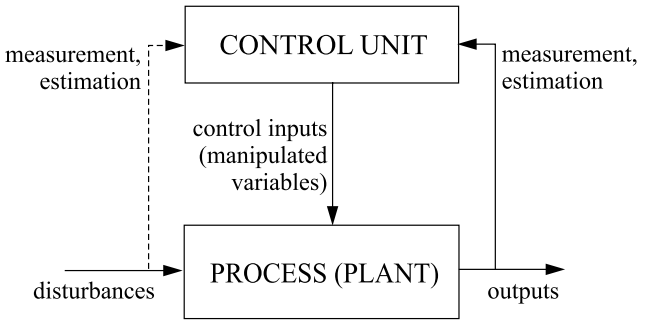


**Fig. 1.1.** General control system structure (reproduced with modifications from Brdys, M.A. and Tatjewski, P., *Iterative Algorithms for Multilayer Optimizing Control*, page 2, copyright 2005 by Imperial College Press, used by permission)

## 1.2 Control Objectives

The control objectives can be of variable nature. For example, the main objective of the control of a passenger airplane is the flight to a defined airport along an assumed flight trajectory. Initially, basic objectives of the control of

technological or economic processes in a market economy are of an economic nature – gaining profit from a production or a commercial activity. Similarly, the initial objective of a telecommunications network control is economic in nature – a long-term profit from the network operation. However, in order to achieve the basic economic objective effectively, it is essential to ensure the realization of a set of partial objectives, which condition the possibility of a safe realization of the basic objective and guarantee the required quality parameters of the offered products or services – all that at a lack of or without complete information about the disturbing process inputs. Moreover, many controlled processes are of a complex nature, with many manipulated and disturbing inputs and many outputs, with mutual interactions between the inputs and outputs. A complex process can be a single reactor or a distillation column. Production lines consisting of several technological processes, mutually influencing each other, are typical examples of very complex processes. Centralized automatic control of such complex processes, although in many cases now theoretically possible, is extremely difficult and is characterized by drawbacks practically eliminating such an option, see *e.g.*, [40, 85, 16]. The most serious of these is the difficulty in ensuring proper safety of the controlled process, difficulty in the necessary participation of people in the process of supervision and reaction to unpredictable phenomena, connected with the necessity of fast and simultaneous processing of large amounts of data. Therefore, in control (and management) of complex processes, there has formed over the years the practice of a *hierarchical* approach, especially a multilayer one, which is a practice supported by theory, see *e.g.*, [78, 40, 16]. The essence of the hierarchical approach is a *decomposition* of the primary, basic task (objective) of the control into a set of partial, less complex and connected tasks, from which every task processes a smaller amount of information and is usually responsible for one partial objective.

There are two basic methods of decomposition of the overall control objective, see *e.g.,* [40, 16] :

- Functional decomposition
- Spatial decomposition

The functional decomposition applies to a process treated as a whole and is based on assigning a set of functionally different partial control objectives – in a structure of vertical, hierarchical dependence, called the *multilayer control structure*. A decision unit connected with each layer makes decisions concerning the controlled process, but each of them makes decisions of a different kind. On the other hand, the spatial decomposition is connected with a spatial structure of a complex (large-scale) controlled process. It is based on a division of the control task or a functionally partial task, *e.g.*, within one layer of the described multilayer structure, into local subtasks of the same functional kind but related to individual spatially isolated parts of the entire complex control process – subtasks of smaller dimensionality, smaller amount of the processed information. This leads to *multilevel structures* [96, 41, 40, 16]. The

subject of interest in this book are multilayer control structures of industrial processes.

The multilayer control structure is a result of a functional decomposition of the general basic control objective. The realization of the basic, economic objective of the on-line control of an industrial (technological) plant can be expressed as the realization of a number of *partial objectives*. The three most important are:

1. Ensuring a safe running of the processes in the controlled plant, *i.e.*, limiting the possibility of emergency situations to an acceptable level.
2. Ensuring required features of the process outputs (quality of products, *etc.*), *i.e.*, maintaining the output variables within ranges of acceptable values.
3. Optimization of effectiveness of the process operation, usually maximization of the product value (under restrictions on usage of raw materials or utilities) or minimization of production costs at an assumed level of production, over a long time horizon.

It is not difficult to notice that the first two partial control objectives are also of an economic nature and they are connected with the basic objective: to maximize the economic effectiveness of the process. The occurrence of failures or other emergency conditions usually leads to serious losses of direct and indirect nature, connected with necessities to remove consequences of delays or production breaks. These losses are usually more severe than the ones resulting from a non-optimal, yet safe production running. Failing to keep to the quality parameters leads, in the best cases, to a partial loss of the profit due to the necessity to lower prices. It may also lead to the loss of the product, if the one not fulfilling the quality requirements cannot find a buyer or is useless for a further production process. Here, financial losses are usually larger than in the case of an economically non-optimal operation of the process, but one which ensures the quality requirements. Let us add that, as a rule, the better the product quality (*e.g.*, cleanliness in a distillation process), the higher the production costs. Therefore, it is usually worth to operate closer to the limits of quality constraints to lower the costs, but this requires more precise control systems because it is more risky due to an ever-present uncertainty connected with the influence of disturbances.

## 1.3 Control Layers

The order in which the three most important partial control objectives are listed in the previous section is not incidental. The most important issue is the safety of the control system, next in the sequence is to care about the quality of the products. Only after ensuring the realization of these two aims, can there be room for on-line economic optimization of variables determining the plant economic objective. Exactly in this order the layers of the basic *multilayer*

*control structure* are located, on top of the controlled process situated at the very bottom, as presented in Fig. 1.2 [40, 16].

The *direct control layer* (called also *basic control layer*, *e.g.*, [11, 115]) is responsible for the safety of dynamic processes. It is usually also equipped with certain control logic responsible for overriding the control designated for normal operating conditions, if violating certain constraints leading to an emergency state is encountered. Only this layer has *direct* access to the plant, and can directly change the values of the control inputs (manipulated inputs), denoted by $u$ in Fig. 1.2. Technical realization of the task of this layer is nowadays ensured, for industrial processes, by *distributed control systems* (DCS). These are complex computer systems of measurement acquisition, control signals generation and on-line process supervision. DCS systems are usually equipped with SCADA (*supervisory control and data acquisition*) type software used for visualization, operator and engineer supervision and archivization of data. Systems of this class were first introduced in the 1970s and

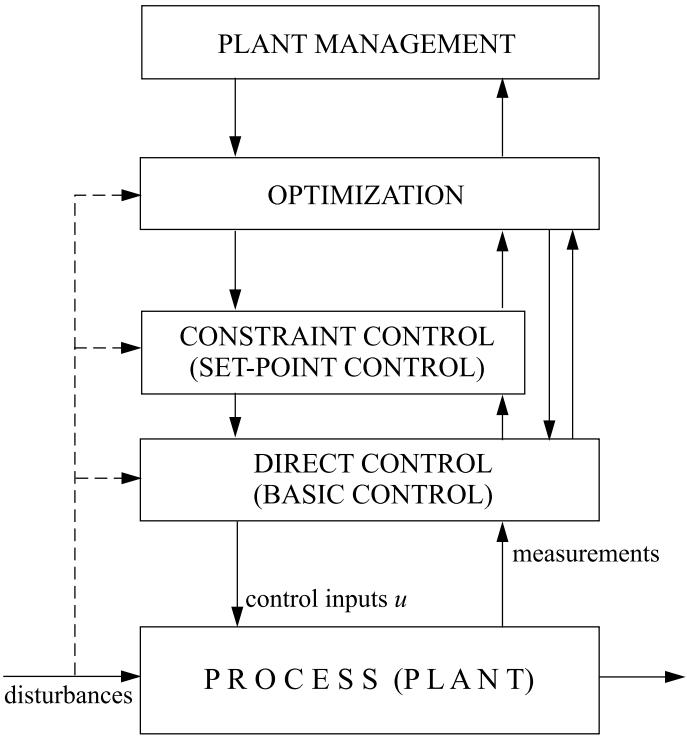


**Fig. 1.2.** Multilayer control structure (reproduced with modifications from Brdys, M.A. and Tatjewski, P., *Iterative Algorithms for Multilayer Optimizing Control*, page 5, copyright 2005 by Imperial College Press, used by permission)

are currently offered by all major vendors on the market. The control tasks for smaller processes can be implemented with the use of *programmable logic controllers* (PLC), or individual multi-function controllers. The distinction between these two classes of equipment is not always clear – more developed versions of PLCs enable the realization of even many control loops, while modern multi-function controllers ensure the possibility of also implementing many logical functions. The most recent, clearly distinguishable trend is the application of personal computers (PCs) to control tasks, usually equipped with specialized cards and software enabling reliable realization of control functions in real time.

Algorithms of direct control should be safe, robust and relatively easy, that is why classic PID algorithms are still dominant. However, computing power of DCS systems, modern PLCs or PCs enables more demanding solutions. Thus, in places where the classic PID control leads to unsatisfactory control quality, more advanced control algorithms can be employed, especially with appropriate modifications of the PID algorithm and, recently, simple realizations of predictive controllers. One can enumerate here the PID control structures with direct correction of the influence of a measured disturbance – feedback-feedforward structures, PID structures with variable gain dependent upon a selected process variable value – gain scheduling, PI controllers with the Smith predictor – for control loops with large delays, nonlinear fuzzy PI or PID algorithms, unconstrained predictive algorithms, adaptive algorithms.

In the literature, one can often find presentation of the *basic control*, defined in this section primarily as the *direct control*, as opposed to the *advanced control* [142]. However, it should be strongly emphasized that the generic feature distinguishing all direct control algorithms is the *direct access* to the controlled process (the process manipulated inputs are outputs of the direct (basic) controllers) and high frequency of intervention (small sampling period) – not the kind of control algorithm employed. Therefore, we shall not keep to this terminology in this book – we shall describe the upper-layer dynamic feedback controllers (control algorithms) with outputs being the set-point values for the direct controllers located below, as the *set-point controllers* constituting the *(dynamic) set-point control layer*, or *constraint controllers* constituting the *(dynamic) constraint control layer* [16]. The latter description is most often used in the literature, see *e.g.*, an excellent industrial review paper [115], because the task of the upper-layer feedback controllers is usually to keep the controlled variables on constraint limits. Both descriptions will be used alternatively throughout the book.

As has been already explained, the output variables of the controllers composing the constraint control layer (set-point control layer) are not the manipulated inputs directly influencing the process, but they are the set-points for the controllers of the direct control layer. Moreover, in control loops of the constraint control layer frequencies of intervention are usually much smaller, *i.e.*, sampling periods are longer – equal to about a minute or longer, while at the direct control layer – about a second at the most, see *e.g.*, [115]. The

objective of the constraint control is to appropriately influence usually slower process variables, which mainly decide on the *production quality parameters*, such as concentrations in reactors or distillation columns. For example, good stabilization (characterized by a small variance of the control error) of the concentration of a key pollutant in a product stream of a distillation process allows to run the process at an operating point located closer to the maximal admissible value of that pollution concentration. The product is then still within the admissible limits but more polluted – and thus cheaper. Therefore, it is required that the constraint control algorithms should be characterized by a high quality of operation (first of all small variance of the control error), they are most frequently applied in cases of multivariable, constrained, nonlinear processes. The most typical, modern solutions applied are the receding horizon model-based predictive control algorithms, commonly described as MPC (Model Predictive Control) algorithms. The most popular were primarily applications based on the usage of the DMC algorithm (Dynamic Matrix Control), developed in the petrochemical industry in the 1970s.

The history and significance of the constraint control layer is directly connected with the development of advanced control algorithms, mainly with applications of the predictive control algorithms. There was no distinction made in any previous literature between the layers of the direct (basic) control and the constraint control (advanced control), see *e.g.*, [78, 40]. It was only the development of the computer technology that enabled the realization of more computationally demanding advanced control algorithms based on process models, such as the DMC algorithm and other predictive control algorithms, and in this way led to a separation of the advanced control layer (set-point control layer, constraint control layer). Since that time this distinction is commonly met in the papers of many leading companies manufacturing control equipment and software, as well as in review papers and basic textbooks, especially those devoted to process control, see *e.g.*, [115, 85, 52, 142].

It should be mentioned that the constraint control layer does not always occur in the control structure. It should not be distinguished in cases when there is no need for the set-point control (constraint control) in the sense described above. Moreover, this layer can not fully separate the direct control layer from the optimization layer – set-point values for a certain part of direct controllers can be directly transmitted from the optimization layer, as it is shown in Fig. 1.2. We shall also not be too rigorous when it is reasonable, in particular including primary controllers of the standard cascade control loops to the direct control layer, although these controllers also act as set-point controllers for the secondary (inner loop) controllers, but they cannot be treated as constraint controllers.

The *optimization layer* is the next, located directly above the direct and constraint control layers, see Fig. 1.2. The objective of its operation is to calculate the *process optimal operating point*, *i.e.*, optimal set-point values for the controllers of directly subordinate feedback control layers. These values usually result from the optimization of an economic objective function which

defines the profit or running costs of the process operation. The optimization problem to be solved is usually a static optimization problem (mathematical programming problem). The *optimal operating point* should be calculated for current values of disturbances (measured or estimated properties of raw materials, utilities, ambient conditions, *etc.*) and varies when these values vary.

The frequency of solving the optimization problem, *i.e.*, the frequency of intervention of the optimization layer is usually much lower than that of the control layers. Moreover, the optimization layer can operate in a synchronous or in an asynchronous mode. In the case of the latter, the optimization task is activated by observed or estimated on-line changes of disturbing process inputs or changes of the required production parameters, transmitted from the top layer of the process management or production planning. Chapter 4 of the book is devoted to algorithms of the optimization layer. The question particularly considered are relations between MPC algorithms of constraint control and steady-state optimization algorithms. Another question of interest is that of the calculation of an optimal operating point in a situation of significant uncertainty revealed by having only an approximate model of the process and/or incomplete information about current values of the disturbances.

It is interesting in recent years to observe the integration of software for predictive constraint control (MPC algorithms) and on-line optimization of the set-points, connected with a rapid development of capabilities of hardware and software for complex control of industrial processes. The MPC algorithms for complex, constrained processes usually operate solving, at every sampling instant, a numerical optimization task. Algorithms of this type require relatively large computing power and a good process model (Chapter 3 is devoted to predictive control algorithms). That is why commercial software packages offering multivariable MPC algorithms are usually complex and expensive, since they usually also contain procedures for modeling and identification of the controlled process, as well as a procedure for on-line optimization of the operating points – directly in the package or in modules closely connected with the package. Massive measurements of process inputs and outputs collected on-line in the DCS can be easily transferred to the higher control layers and used in algorithms for model identification (tuning, adaptation) as well as for constraint control and optimization. The optimization procedure supplies the feedback constraint control algorithms with appropriate values for the controlled outputs; it is activated in an adequate way which is tuned to the entire process operation. An interesting case, from the point of view of the integration of the constraint control and optimization, is a case when the possible number of outputs of the predictive controller (its "manipulated variables") can be larger than the number of the associated controlled process outputs.

The highest layer presented in Fig. 1.2 is the *plant management* (or *production planning*) layer. Its task is to establish operating conditions for the optimization layer, *i.e.*, production goals and parameters – an economic objective function and constraints. This layer operates on the brink of the process economic environment, reacting accordingly to orders concerning an assort-

ment and amount of production, prices, sales, *etc.* – coming directly from the market environment or larger plant environment of which the controlled process is an element. Frequency of intervention of this layer can correspond to a period of a production shift, or even to several days. Algorithms of its operation and employed process models are beyond the scope of this book.

The basic features distinguishing the individual control layers are separate, *isolated control objectives and different intervention frequencies*. Table 1.1 lists the basic tasks of individual layers of an industrial process control structure and their typical intervention periods, see also *e.g.*, [52].

Finishing the description of basic elements of the multilayer control structure, let us point out that *the basic reason* of its importance is the following:

> *A division (decomposition) of the initial overall control problem into several simpler, related subproblems simplifies the process of design, control and supervision – simpler control systems are designed for the particular layers realizing partial control goals, not the one complex centralized control system for the entire process.*

Therefore, for complex processes the multilayer approach is not only the most practical and effective, but often the only one possible.

**Table 1.1.** Basic tasks and intervention periods of control layers

| **Control layer** | **Basic task** | **Typical period of intervention** |
|---|---|---|
| direct control (basic control) | process stabilization (safe operation) | fraction of a second, second |
| constraint control (set-point control) | quality control – advanced feedback control of key variables (often close to constraints) | minute, minutes |
| optimization | maximization of running economic effects | hour, hours |
| production management | maximization of economic effects for longer periods | shift, day, several days |

## 1.4 Process Modeling in a Multilayer Structure

The multilayer control structure consisting of regulatory control and optimization layers, as presented in Fig. 1.3, will now be considered in more detail,

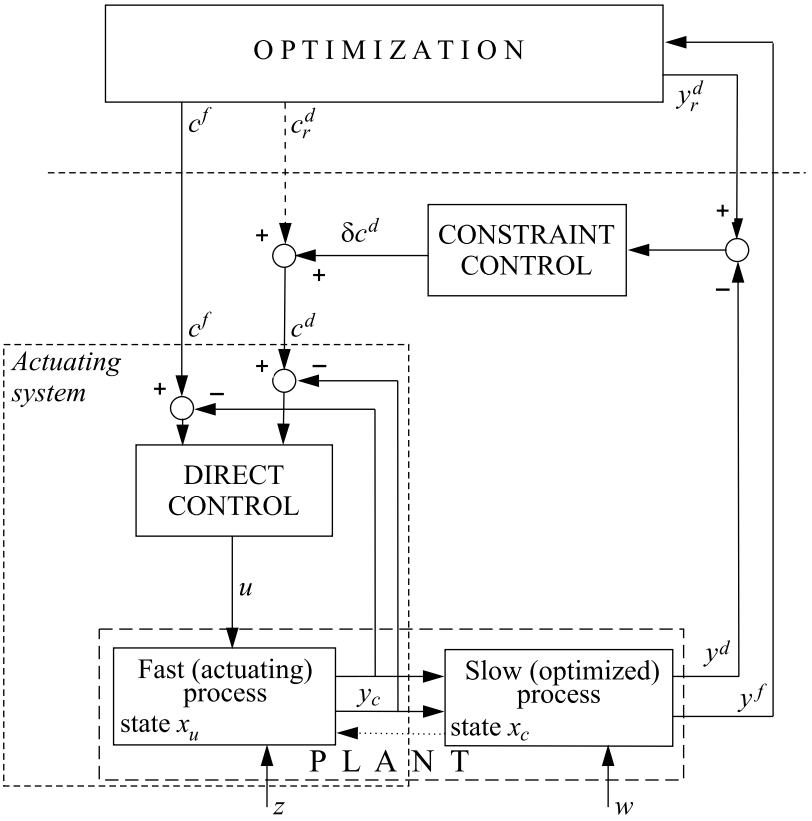


**Fig. 1.3.** Multilayer structure of control and optimization with decomposition of the plant dynamics (reproduced with modifications from Brdys, M.A. and Tatjewski, P., *Iterative Algorithms for Multilayer Optimizing Control*, page 12, copyright 2005 by Imperial College Press, used by permission)

see also [35, 16]. A characteristic feature of this structure is a representation of the plant dynamics in a form decomposed into a cascade of processes (subprocesses) with a faster and a slower dynamics. The "fast process" is influenced by control inputs $u$ and by fast changing disturbances $z$. The output variables of this part of the plant, denoted by $y_c$, are inputs to the part characterized by slower dynamics, which are directly influenced by slower disturbances $w$. The output vector $y = (y^f, y^d)$ consists of variables significant for the tasks of economic constrained optimization of the controlled process. The objective of the constraint control presented in Fig. 1.3 is to cause that certain elements of the plant output vector $y$, denoted as a sub-vector $y^d$, to be kept on values $y_r^d$ prescribed by the constrained optimization, *i.e.*, to enforce the equality constraint

$$y^d(t) = y_r^d$$

This corresponds to a practical case of a feedback control of important constraints on certain output variables. An example of that is a feedback control of the concentration of a key pollutant in a product stream of a distillation column, keeping it on a value which is safely close to the maximum admissible one (the maximal value of the pollutant concentration cannot be approached too closely due to an unavoidable variance of the control error and a resulting threat of exceeding the limit). The remaining, uncontrolled (free) components of the process output vector $y$ are denoted as a sub-vector $y^f$.

The key role in the presented plant decomposition is played by a correct selection of the *controlled output variables* $y_c$. The set-points for these controlled variables, denoted by $c = (c^f, c^d)$ are decision variables in the optimization problem. The choice of the controlled variables is usually a result of an experience of designers and operators of a controlled process and depends on a formulation of the optimization task. It should be such as to ensure the following:

- Stabilization of $y_c$ on reasonably selected values of its set-point $c$ should ensure safe control of the process, *i.e.*, should uniquely define the values of significant elements of its state vector.
- Values of the set-point $c$, being decision variables of the optimization task, should allow for realization of this task. This means that they should ensure full usage of the possibilities to influence those process output variables which are significant for an improvement of the value of the optimized criterion (the objective function) and for values of the constraints.

It should be emphasized that the former of these conditions always has to be satisfied; it is the basic requirement for a correct, safe operation of the direct control layer. If a choice of the controlled variables satisfying also the second condition does not ensure a satisfactory realization of the optimization goals, then the set of these variables is not properly chosen or too limited. In this case, one should pose less ambitious optimization goals, or enrich the process control structure with additional manipulated variables $u$ and corresponding additional controlled outputs $y_c$ – in this way allowing for more possibilities in satisfactory realization of the optimization goals (in normal operating conditions the number of controlled outputs should not be higher than that of manipulated variables, see [40, 39]).

Let us consider the multilayer structure presented in Fig. 1.3. If direct control systems are operating properly, then, apart from time periods directly following fast (step) changes of the set-points $c$ or disturbances, we can assume that the following is true

$$y_c(t) = c(t) \tag{1.1}$$

It can then be assumed that, from a point of view of the constraint controllers and optimization algorithms, only slower dynamics of the plant can be

taken into account. An input-output relation defining these dynamics can be described by an operator $F$,

$$y(t) = F(c(t), w(t)) \tag{1.2}$$

Therefore, for the constraint controllers and optimization algorithms, the fast process along with the direct control layer can be treated as an *actuating system* – which enforces the set-point values $c(t)$ of the controlled variables $y_c(t)$, *i.e.*, enforces the equality $y_c(t) = c(t)$. The term "actuating system" [35] has been introduced by analogy to an "actuating element" which is, for example, a valve with a positioner. That is why the fast process in Fig. 1.3 was named an *actuating process*, while the slow process is called an *optimized process*. Because the plant behavior characterized only by this process is seen by the upper layers, especially by the optimization layer.

An analytical formula of the plant model operator (1.2) is rarely available. However, it is implicitly given by the following model typically assumed for continuous systems with lumped parameters

$$\begin{aligned} \frac{dx_c(t)}{dt} &= f_c(x_c(t), c(t), w(t)) \\ y(t) &= g_c(x_c(t), c(t)) \end{aligned} \tag{1.3}$$

where $x_c$ is a state vector of the slow process, see Fig. 1.3, and the equality $y_c(t) = c(t)$ was consequently assumed (*i.e.*, ideal operation of the actuating system was assumed), in this way eliminating from the model variables $y_c(t)$.

A description of the entire plant dynamics can be assumed, analogously, in the following general form

$$\begin{aligned} \frac{dx_u(t)}{dt} &= f_1(x_u(t), x_c(t), u(t), z(t), w(t)) \\ \frac{dx_c(t)}{dt} &= f_2(x_u(t), x_c(t), u(t), w(t)) \\ y(t) &= g(x_u(t), x_c(t), u(t)) \end{aligned} \tag{1.4}$$

where the state vector $x(t)$ was written in a divided form corresponding to the fast and slow states, $x(t) = (x_u(t), x_c(t))$, and consequently a lack of direct influence of fast changing disturbances $z(t)$ on the sub-vector of the slow state $x_c(t)$ was assumed. A decomposition, namely a division of the whole state vector $x$ into the sub-vectors of "fast" and "slow" states, $x_u$ and $x_c$, is in each case an individual question resulting from process characteristics and requirements concerning the controlled variables described above. Models (1.4) and (1.3) should of course be completed by a set of appropriate initial conditions, essential for formal analytical considerations or any numerical calculations.

Assuming the equation defining the controlled variables is in the form of a function of the process states and inputs [40, 39]

$$y_c(t) = h(x_u(t), u(t)) \tag{1.5}$$

one can consider relations between models (1.4) and (1.3). Assuming (1.1) holds, *i.e.*, $c(t) = y_c(t)$, the following result is obtained

$$\begin{aligned}\frac{dx_c(t)}{dt} &= f_c(x_c(t), h(x_u(t), u(t)), w(t)) \\ &= f_2(x_u(t), x_c(t), u(t), w(t))\end{aligned} \tag{1.6}$$

and similarly

$$\begin{aligned}y(t) &= g_c(x_c(t), h(x_u(t), u(t))) \\ &= g(x_u(t), x_c(t), u(t))\end{aligned} \tag{1.7}$$

In the multilayer structure each layer controls in fact the same plant, but each one does it in a different way. The direct control layer is the only one with direct access to the process manipulated inputs. It obtains measurements of all available output variables which are significant for the stabilization and safe operation of the plant, first of all the measurements of the output variables which change faster and decide on the possibility of a quick reaction of direct feedback control systems. The constraint control layer which intervenes and obtains measurements more rarely perceives the process in a different way. It sees it together with the direct control systems, for which it assigns decisions in the form of the set-point values. Moreover, when the direct controllers operate sufficiently quickly and precisely, then the fast-changing transients induced by the influence of fast-changing disturbances are not significant for the constraint control layer and can be ignored. Therefore, the constraint control layer deals then with a different "plant", whose dynamics is determined by the plant processes with slower dynamics. Similarly, the optimization layer perceives the plant along with all subordinate feedback control systems. Therefore, modeling the plant for control purposes with the aim of capturing only basic dependencies significant to the control design at a given layer is different at each layer, leading to different models. Table 1.2 presents typical, basic features of models at particular layers.

**Table 1.2.** Models of the controlled plant at different control layers

| **Control layer** | **Typical model** |
|---|---|
| direct control<br>(basic control) | fast dynamic<br>(linear, rarely nonlinear) |
| constraint control<br>(set-point control) | slow dynamic<br>(linear, nonlinear) |
| optimization | nonlinear static<br>(rarely dynamic) |
| production management | linear aggregate<br>(balance based) |

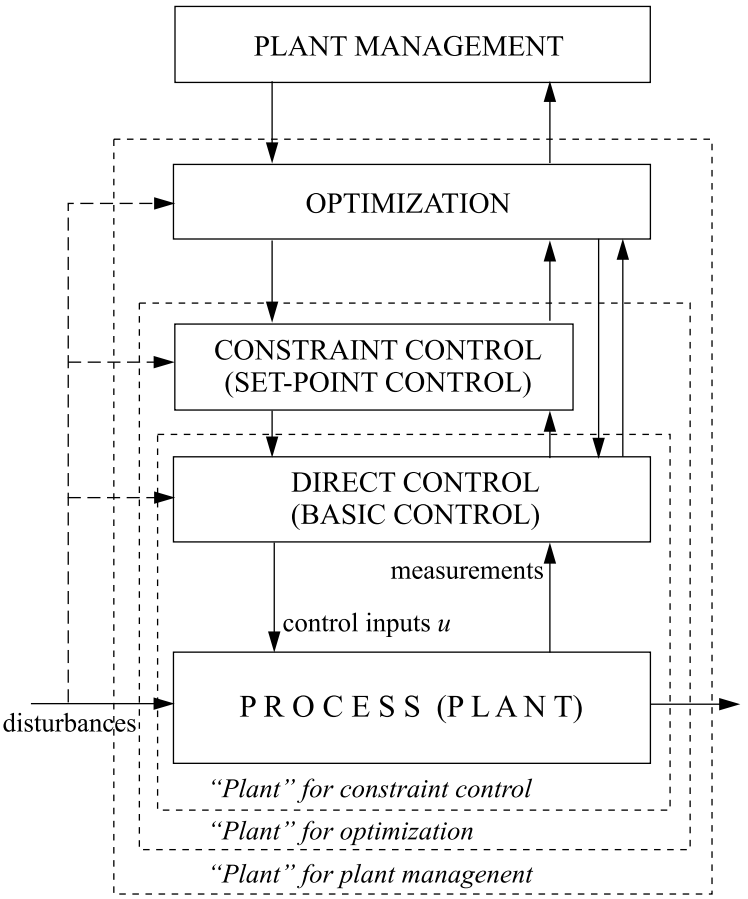


**Fig. 1.4.** Modeling of the controlled plant in a multilayer structure

Let us note that the higher the layer, the less detailed and more aggregated the model used, possessing slower dynamics, or even static. For the considered control layer, the plant with all the lower layers is a certain "actuating system", which should enforce decisions of this layer. For example, in Fig. 1.3 the "actuating system", consisting of the plant together with the direct control systems, forces the controlled variables $y_c$ to keep to the values $c = (c^d, c^f)$. Figure 1.4 presents plant modeling at different control layers, clearly showing the characteristic feature of "nesting". The plant is modeled at a given layer together with all the control systems of the lower layers – along with consequences of their operation enabling appropriate simplification and aggregation adequate to the task and time scale of the layer. This is, of course, a simplified approach, but the one proven in engineering practice, enabling efficient design and on-line operation of the process control and optimization.

**Example 1.1**
Decomposition of a process model and a set-point control will be illustrated by a simple example of a continuously-stirred tank reactor (CSTR), presented in Fig. 1.5. The inflow to the reactor is a stream of component $A$ with a flow

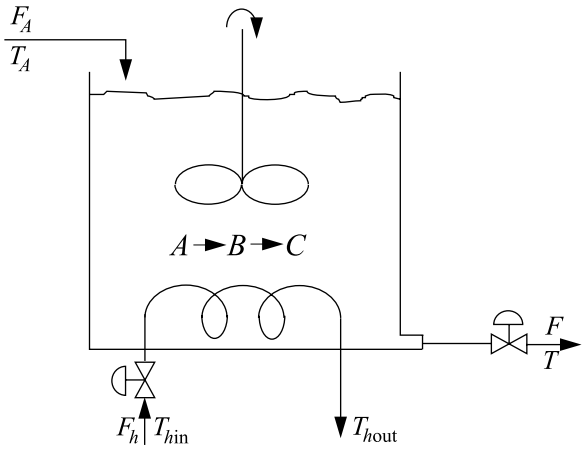


**Fig. 1.5.** Continuously-stirred tank reactor (CSTR), Example 1.1

rate $F_A$ $[kg/min]$ and a temperature $T_A$ $[K]$. Two reactions take place in the tank: $A \rightarrow B \rightarrow C$ with reaction rates $r_B[\frac{1}{min}]$ and $r_C[\frac{1}{min}]$, respectively. The first reaction is endothermic and the second one is exothermic. The mixture is heated by a heating medium flowing through a pipe heat exchanger located in the bottom part of the reactor. The flow rate of the heating medium $F_h$ and the product outflow rate from the reactor $F$ can be controlled by appropriate valves.

The following simplifying assumptions are taken when modeling the process:

1. A perfect mixing in the tank is assumed, therefore the concentrations, $C_A$ of component A and $C_B$ of component B, and the temperature $T$ of the mixture are the same in the entire tank volume.
2. The reaction rate is described by the following models

$$r_B = k_1(T)\, C_A = k_{10} \exp(-\frac{E_1}{RT})\, C_A \tag{1.8}$$

$$r_C = k_2(T)\, C_B = k_{20} \exp(-\frac{E_2}{RT})\, C_B \tag{1.9}$$

where

$$k_{10} = 9000 \ll k_{20} = 35000\ [\frac{1}{min}]$$
$$\frac{E_1}{R} = 4000 < \frac{E_2}{R} = 5200\ [K]$$

while reaction heats are known and are $h_1\ [\frac{J}{kg}]$ and $h_2\ [\frac{J}{kg}]$, respectively.

3. The influence of the shaft work and the heat exchange with the environment are negligible. Moreover, the same mass densities $\rho\ [\frac{kg}{m^3}]$ and heat capacities $c_w\ [\frac{J}{K\ kg}]$ of all mixture components and of the mixture itself are assumed – to simplify the process modeling.
4. The mean temperature $T_{h\mathrm{m}}$ of the heating medium at the input and at the output of the heat exchanger pipe is taken as a driving force of the heat exchange. The heat transfer through the surface of the exchanger is assumed to be described by the following empirical formula

$$H = a(F_h)^b(T_{h\mathrm{m}} - T) \tag{1.10}$$

where $a$ and $b$ are coefficients resulting from the construction of the heat exchanger [85].

5. The range of variation of the flow rate $F_h$ enables to reach and stabilize the reactor temperature within $300 \div 360\ K$.

With the assumptions presented above the reactor can be treated as a process characterized by four state variables describing mass balances of the mixture and all its components and a heat balance. For modeling purposes we shall assume the following *state variables*:

$W$ – mass of the mixture in the tank,
$C_A$ – concentration of component $A$,
$C_B$ – concentration of component $B$,
$T$ – temperature in the tank.

The *state equations* can be formulated as follows (dependence on time $t$ is omitted in the following equations, to simplify the notation):

– for $W$ (the mass balance in the tank):

$$\frac{dW}{dt} = F_A - F$$

– for $C_A$ (the mass balance of component $A$):

$$\begin{aligned}\frac{d(WC_A)}{dt} &= W\frac{dC_A}{dt} + C_A\frac{dW}{dt} \\ &= F_A - FC_A - Wk_1(T)C_A \\ W\frac{dC_A}{dt} &= -C_A(F_A - F) + F_A - FC_A - Wk_1(T)C_A \\ &= -C_AF_A + F_A - Wk_1(T)C_A\end{aligned}$$

thus

$$\frac{dC_A}{dt} = \frac{1 - C_A}{W} F_A - k_1(T) C_A$$

– for $C_B$ (the mass balance of component $B$):

$$\begin{aligned} \frac{d(WC_B)}{dt} &= W\frac{dC_B}{dt} + C_B\frac{dW}{dt} \\ &= -FC_B + Wk_1(T)C_A - Wk_2(T)C_B \\ W\frac{dC_B}{dt} &= -C_B(F_A - F) - FC_B + Wk_1(T)C_A - Wk_2(T)C_B \\ &= -C_B F_A + Wk_1(T)C_A - Wk_2(T)C_B \end{aligned}$$

thus

$$\frac{dC_B}{dt} = -\frac{C_B}{W} F_A + k_1(T)C_A - k_2(T)C_B$$

– for temperature $T$ (the heat balance):

$$\begin{aligned} &\frac{d(c_w WT)}{dt} = c_w W\frac{dT}{dt} + c_w T\frac{dW}{dt} = \\ &= c_w F_A T_A - c_w FT + F_h c_h(T_{h\text{in}} - T_{h\text{out}}) - h_1 Wk_1(T)C_A + h_2 Wk_2(T)C_B \end{aligned}$$

$$\begin{aligned} c_w W\frac{dT}{dt} = -c_w TF_A + c_w F_A T_A + F_h c_h(T_{h\text{in}} - T_{h\text{out}}) - h_1 Wk_1(T)C_A + \\ + h_2 Wk_2(T)C_B \end{aligned}$$

From among all variables occurring on the right hand side of the above equation, the output temperature of the heating medium $T_{h\text{out}}$ is a variable fully determined by the input variables. Considering the simplifying assumption 4 concerning the heat exchange together with the formula (1.10), the temperature $T_{h\text{out}}$ can be eliminated from the heat balance equation, because there are two dependencies for the heat transfer:

$$H = a(F_h)^b\left(\frac{T_{h\text{in}} + T_{h\text{out}}}{2} - T\right)$$

$$H = F_h c_h(T_{h\text{in}} - T_{h\text{out}})$$

Evaluating $T_{h\text{out}}$ from the second equation

$$T_{h\text{out}} = -\frac{H}{F_h c_h} + T_{h\text{in}}$$

and substituting to the first one we get

$$\begin{aligned} H &= a(F_h)^b\left(\frac{1}{2}\left(T_{h\text{in}} - \frac{H}{F_h c_h} + T_{h\text{in}}\right) - T\right) \\ &= a(F_h)^b(T_{h\text{in}} - T) - \frac{a}{2}(F_h)^b\frac{H}{F_h c_h} \end{aligned}$$

This equation allows us to determine the following formula for the exchanged heat $H$

$$H = \frac{a(F_h)^{b+1}}{F_h + \frac{a(F_h)^b}{2c_h}}(T_{h\text{in}} - T) \tag{1.11}$$

Therefore, the equation describing the heat transfer can be formulated in the form

$$c_w W \frac{dT}{dt} = c_w F_A(T_A - T) + H(T_{h\text{in}} - T, F_h) - h_1 W k_1(T) C_A + h_2 W k_2(T) C_B$$

or, equivalently,

$$\frac{dT}{dt} = \frac{F_A}{W}(T_A - T) + \frac{1}{c_w W} H(T_{h\text{in}} - T, F_h) - \frac{h_1}{c_w} k_1(T) C_A + \frac{h_2}{c_w} k_2(T) C_B$$

where the functional dependence $H(T_{h\text{in}} - T, F_h)$ is given by (1.11).

In conclusion, the system of state equations of the considered reactor takes the following form

$$\frac{dW}{dt} = F_A - F \tag{1.12a}$$

$$\frac{dT}{dt} = \frac{F_A}{W}(T_A - T) + \frac{1}{c_w W} H(T_{h\text{in}} - T, F_h) - \frac{h_1}{c_w} k_1(T) C_A + \frac{h_2}{c_w} k_2(T) C_B \tag{1.12b}$$

$$\frac{dC_A}{dt} = \frac{1 - C_A}{W} F_A - k_1(T) C_A \tag{1.12c}$$

$$\frac{dC_B}{dt} = -\frac{C_B}{W} F_A + k_1(T) C_A - k_2(T) C_B \tag{1.12d}$$

where reaction rates $k_1(T)$ and $k_2(T)$ are given by (1.8) and (1.9), while the amount of the heat delivered $H(T_{h\text{in}} - T, F_h)$ by (1.11).

The order in which the state equations (1.12a)-(1.12d) were positioned is not incidental. It follows from the nature of the phenomena occurring in the reactor, where the state variables $W$ and $T$ change faster than concentrations $C_A$ and $C_B$, when affected by changes of the manipulated or disturbing inputs, *i.e.*, $F$ and $F_h$ or $F_A$, $T_A$ and $T_{h\text{in}}$, respectively. Moreover, stabilization of $W$ and $T$ ensures safe operation of the reactor, *i.e.*, without over-filling or excessive emptying of the tank and without exceeding the admissible temperature. Thus, decomposition of the state vector, according to (1.4), would be:

$$x_u = [W\ T]^T$$
$$x_c = [C_A\ C_B]^T$$

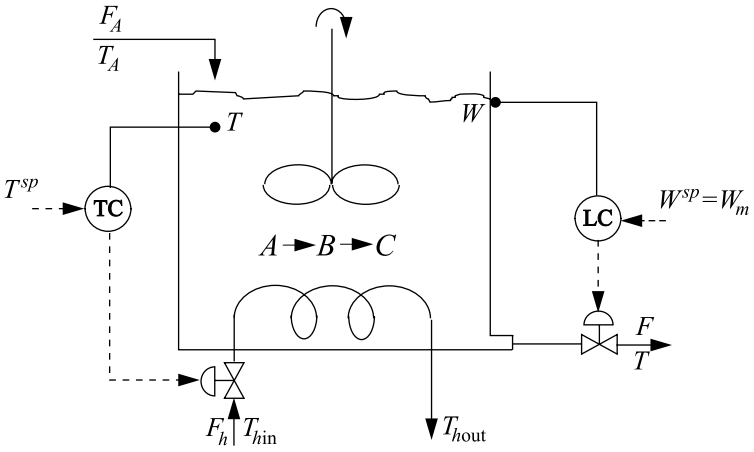


**Fig. 1.6.** CSTR with direct controllers (LC - level control, TC - temperature control)

The vector of the manipulated variables $u$ is

$$u = [F\ F_h]^T$$

while the faster changing state variables, *i.e.*, $T$ and $W$, can be taken as process outputs controlled by the direct control layer,

$$y_c = [T\ W]^T$$

Figure 1.6 presents the reactor together with direct control loops of level and temperature. The set-point values for these direct control loops, in the terminology of Fig. 1.3, are the components $T^{sp}$ and $W^{sp} = W_m$ of the vector $c$

$$c = [T^{sp}\ W_m]^T$$

For a design of the set-point control (constraint control) layer an ideal operation of direct controllers is assumed, *i.e.*,

$$W(t) = W_m$$
$$T(t) = T^{sp}(t)$$

where particularly the former of these equalities can be relatively accurately enforced (fast stabilization of the level by manipulating the outflow rate $F$). With these assumptions made, the dynamics of the slow process (subprocess), see (1.3), can be described by the following equations

$$\frac{dC_A(t)}{dt} = \frac{1 - C_A(t)}{W_m} F_A(t) - k_1(T) C_A(t) \tag{1.13}$$

$$\frac{dC_B(t)}{dt} = -\frac{C_B(t)}{W_m} F_A(t) + k_1(T) C_A(t) - k_2(T) C_B(t) \tag{1.14}$$

where $T = T(t) = T^{sp}(t)$.

Let us consider the following formulation (a) of the *control objective* (further on we shall discuss another formulation which will be marked with (b)):

(a) *Stabilization of the concentration $C_B$ of the reactant $B$ in the product stream at the value $C_B^{sp} = 0.25$, assuming a constant, maximal filling of the tank $W(t) = W_m = const.$ and slow changing fluctuations of the inflow rate resulting in the residence time varying within the limits $W_m/F_A = 25 \pm 5$ $[min]$. It is also assumed that the concentration $C_B$ is measured on-line by an analyzer, however with a measurement time much longer than the control interval (sampling period) of the direct controllers of level and temperature.*

Realization of the above objective assumes keeping the controlled process in a steady-state – the concentration $C_B$ should be stabilized. Process (1.13)-(1.14) is in a steady-state when

$$\frac{dC_A(t)}{dt} = 0, \ \frac{dC_B(t)}{dt} = 0,$$

in spite of an influence of slowly changing disturbance $F_A(t)$. The equations defining the steady-state model of the slower part of the reactor are:

$$0 = \frac{1 - C_A}{W_m} F_A - k_{10} \exp(-\frac{E_1}{RT})\, C_A$$

$$0 = -\frac{C_B}{W_m} F_A + k_{10} \exp(-\frac{E_1}{RT})\, C_A - k_{20} \exp(-\frac{E_2}{RT})\, C_B$$

Evaluating $C_A$ from the first equation

$$C_A = \frac{\frac{F_A}{W_m}}{\frac{F_A}{W_m} + k_{10} \exp(-\frac{E_1}{RT})}$$

and substituting to the second one we obtain

$$C_B \left( \frac{F_A}{W_m} + k_{20} \exp(-\frac{E_2}{RT}) \right) = k_{10} \exp(-\frac{E_1}{RT}) \frac{\frac{F_A}{W_m}}{\frac{F_A}{W_m} + k_{10} \exp(-\frac{E_1}{RT})}$$

$$C_B = \frac{\frac{F_A}{W_m} k_{10} \exp(-\frac{E_1}{RT})}{\left( \frac{F_A}{W_m} + k_{10} \exp(-\frac{E_1}{RT}) \right) \left( \frac{F_A}{W_m} + k_{20} \exp(-\frac{E_2}{RT}) \right)}$$

$$C_B = \frac{\frac{W_m}{F_A} k_{10} \exp(-\frac{E_1}{RT})}{\left( 1 + \frac{W_m}{F_A} k_{10} \exp(-\frac{E_1}{RT}) \right) \left( 1 + \frac{W_m}{F_A} k_{20} \exp(-\frac{E_2}{RT}) \right)} \tag{1.15}$$

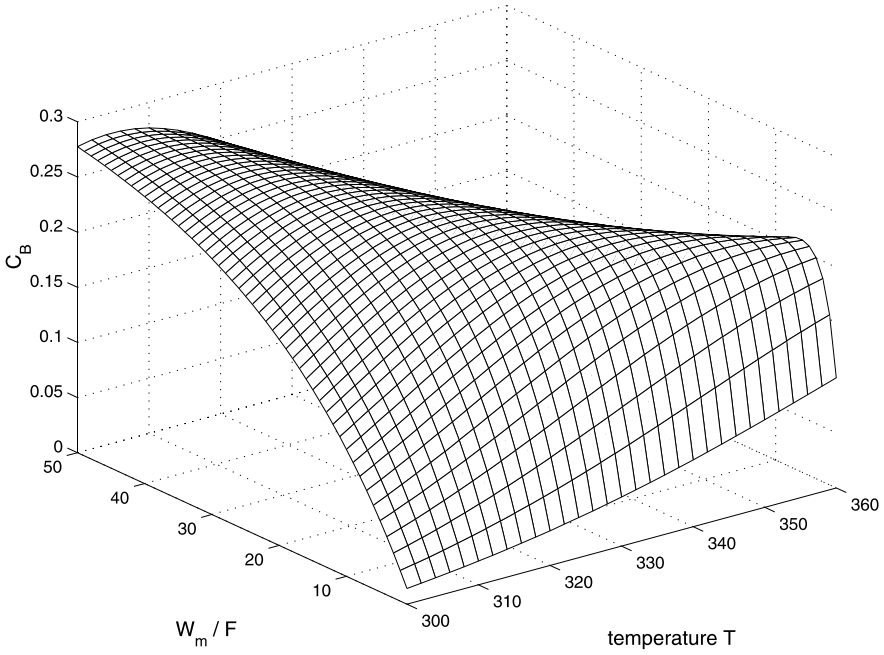


**Fig. 1.7.** Surface $C_B(T, \frac{W_m}{F})$

The formula (1.15) describes a steady-state model, namely the static characteristics of the concentration $C_B$,

$$C_B = C_B(T, F_A) \tag{1.16}$$

in the process described by (1.13)-(1.14). In a general terminology of the multilayer control structure, (1.16) defines the function $y = F(c, w)$ for our example CSTR problem, compare with (1.2).

The mapping (1.16) is easier to present in a slightly different coordinate system, namely taking $\frac{W_m}{F_A}$ instead of $F_A$, because the residence time $\frac{W_m}{F_A}$ (the time of filling the tank to the full contents $W_m$ by a constant inflow $F_A$) is well interpretable and widely used. The shape of the surface $C_B(T, \frac{W_m}{F_A})$, evaluated for numerical values given at the beginning of this example problem formulation, is shown in Fig. 1.7, whereas level sets of this surface are shown in Fig. 1.8. It is assumed in these figures that $F = F_A$, which results from the assumed constant filling of the tank, $W(t) = W_m$.

The formulated control objective based on a stabilization of the concentration $C_B$ can be implemented by an upper-layer controller (constraint controller), as it is shown in Fig. 1.9. Let us note that using the supervisory feedback control follows not only from slower dynamics of the concentration, but it is

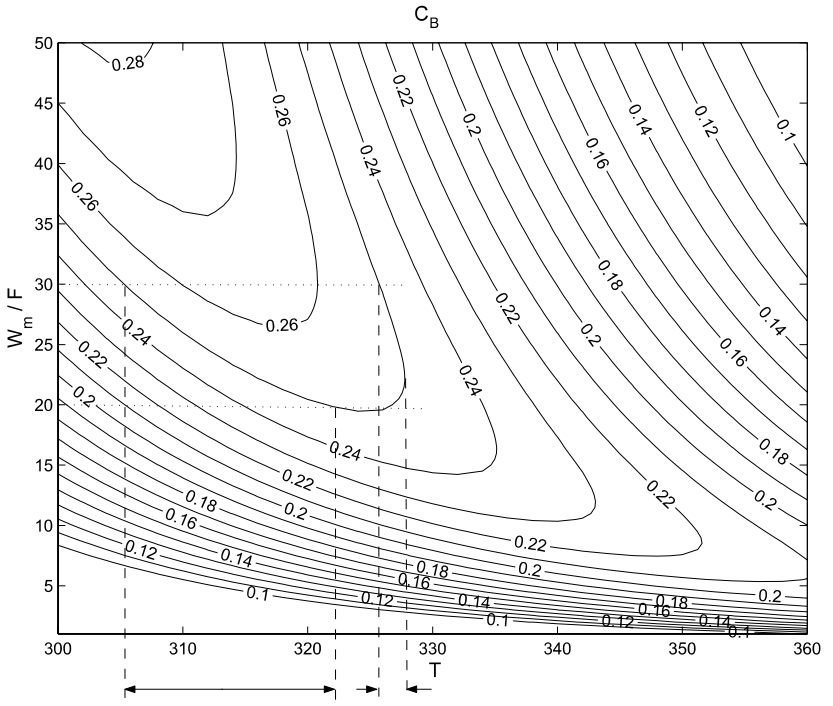


**Fig. 1.8.** Level sets of the surface $C_B(T, \frac{W_m}{F})$, with marked sets of possible temperature values being the output of the controller stabilizing $C_B$

also forced by the time of its measurement, which is much longer than the sampling period of the level and temperature controllers. This measurement time defines a lower limit for the sampling period of the constraint controller.

Since dynamics of the direct control system of level and temperature is faster than the dynamics described by the model (1.13)-(1.14), then it is possible to use this model only (which is much simpler than the model of the dynamics of the entire plant) when designing the concentration controller. This, however, should be followed by a verification of the behavior of the entire two-layer control system, *i.e.*, by a simulation or by a real time experiment in the controlled plant – to correct the settings of the controller, if needed. Let us note that the described control of the concentration $C_B$ is in fact in a cascade control structure, but with different scan times in primary and secondary loops (a multi-rate control system). Figure 1.10 presents a diagram of the control system from Fig. 1.9 in the language of multilayer structure blocks from Fig. 1.3, with a decomposed description of the process.

It can be seen from the shape of level sets of the surface $C_B(T, \frac{W_m}{F})$ that the controlled process is nonlinear and the same values of the concentration can be achieved at different temperature values. Fig. 1.8 presents two different

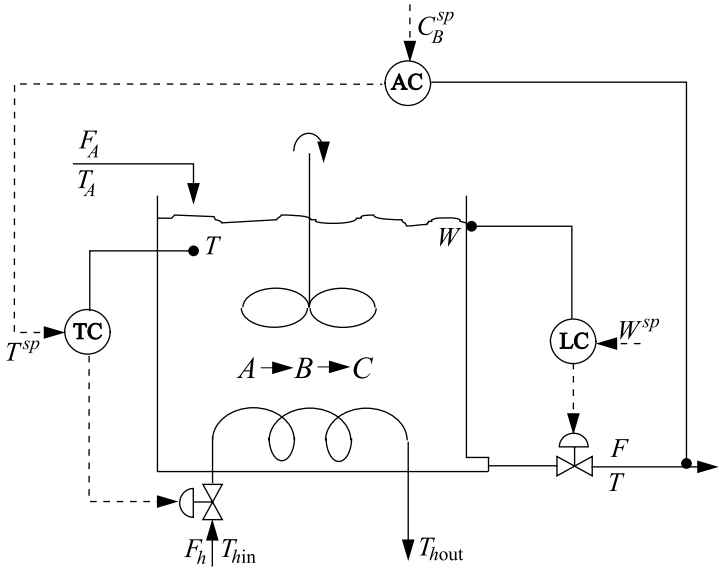


**Fig. 1.9.** CSTR with direct control loops (LC and TC) and constraint control (AC) for stabilization of $C_B$

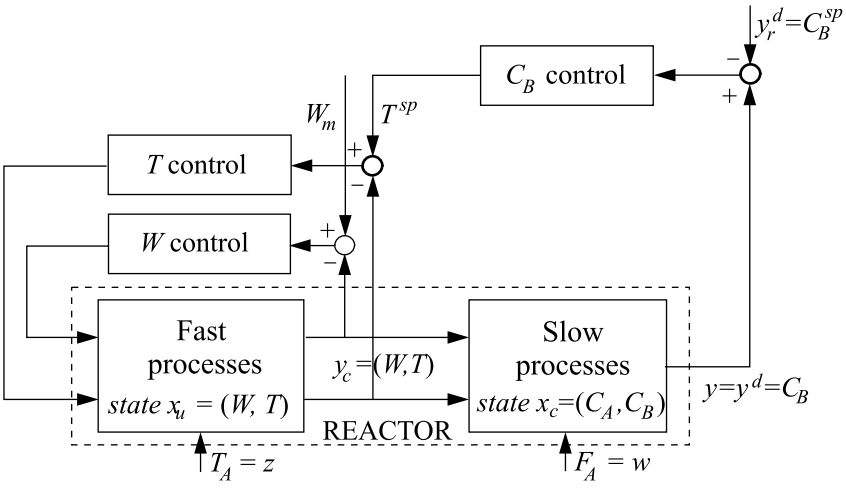


**Fig. 1.10.** Decomposed block diagram of the CSTR multilayer control system for $C_B$ control

and separate ranges of the temperature $T$ stabilizing the concentration on the same value 0.25. The analysis indicating which of these ranges is more suitable for a controller operation and how to design a controller in this situation, is left to the reader. Let us mention only that when applying a standard linear controller, *e.g.*, a PID controller, we usually first identify a linear process model at a chosen operating point and then controller settings are tuned using this model. Certainly, a different operating point satisfying $\frac{W_m}{F_A} = 0.25$ and a different linear model will correspond to each of the two temperature ranges presented in Fig. 1.8. □

## 1.5 Optimization Layer

The task of the *optimization layer* is to select optimal values of set-points for the feedback controllers of lower control layers, optimizing a defined objective function of economic nature. Generally, optimal state of a process can be dynamic and the process can be operated in a dynamic mode. In this situation, the optimization layer evaluates optimal dynamic trajectories of the controlled variables, as trajectories of the set-points for the feedback controllers. These result from solving the constrained dynamic optimization problem. The *objective function* (*performance function*) of such a problem can be formulated as follows:

$$J(c,y) = \int_{t_0}^{t_k} Q(c(t),y(t))dt \tag{1.17}$$

where $t_k - t_0$ is an optimization horizon. The objective function is optimized with respect to several constraints:

- Equations of process dynamics (1.3),
- Inequality constraints, of physical and technological origin, on decision variables $c$ and output values $y$. These constraints can be written in a general form, defined by certain functions $g$ and $h$,

$$c(t) \in C = \{c(t) : \ g(c(t)) \leq 0\}, \quad t \in [t_0 \ t_k] \tag{1.18}$$

$$y(t) \in Y = \{y(t) : \ h(y(t)) \leq 0\}, \quad t \in [t_0 \ t_k] \tag{1.19}$$

  In practice, these are mainly constraints on instantaneous maximum or minimum values of particular variables,

$$c(t) \in C = \{c(t)\text{: } c(t) \geq c_{\min}, \ c(t) \leq c_{\max} \}, \quad t \in [t_0 \ t_k]$$

$$y(t) \in Y = \{y(t)\text{: } y(t) \geq y_{\min}, \ y(t) \leq y_{\max} \}, \quad t \in [t_0 \ t_k]$$

  We have not considered here constraints on state variables $x_c$, as this can always be done by assuming that constrained components of the vector $x_c$ are a part of the vector of outputs. This assumption seems reasonable since for a constraint control these variables must be directly or indirectly measured, thus they should be treated as outputs.

Using the introduced dependencies, the task of the *dynamic optimization* is to optimize the objective function (1.17) subject to equality constraints (1.3) and inequality constraints (1.18), (1.19). In a case of a minimization, it is in the form

$$\begin{aligned} \min\{\ J(c,y) &= \int_{t_0}^{t_k} Q(c(t),y(t))dt\ \} \\ \text{subj. to:}\ & dx_c(t)/dt = f_c(x_c(t), c(t),w(t)) \\ & y(t) = g_c(x_c(t), c(t)) \\ & c(t) \in C \\ & y(t) \in Y, \quad t \in [t_0\ t_k] \end{aligned} \tag{1.20}$$

Generally, it is either a problem with a periodic solution – then the period determines the optimization horizon, or a problem with a horizon resulting from the dynamics of disturbances $w(t)$ and the dynamics of the process itself. In the latter case the formulated optimization problem is usually an element of a control in a repetitive structure with a receding horizon. However, when controlling continuous industrial processes, the disturbances usually do not allow for long-term forecasts, thus optimal solutions stabilize as constant or periodic.

A *steady-state control mode* dominates in industrial practice, that is a control mode when set-point values for the process feedback controllers are kept constant during certain periods of time. Optimal constant set-point values are obtained through the solution of the *steady-state (static) optimization problem* which takes the form

$$\begin{aligned} & \min Q(c,y) \\ \text{subj. to:}\ & 0 = f_c(x_c, c, w) \\ & y = g_c(x_c, c) \\ & c \in C \\ & y \in Y \end{aligned} \tag{1.21}$$

In this problem formulation actual (measured or estimated) disturbance values $w$ are used as constant parameters. However, the disturbances are usually varying, more or less slowly, or changing rarely but abruptly. Therefore, the set-point values should be adapted to these changes, to keep the process close to optimal operation. This is done by solving the optimization problem (1.21) at regular time intervals corresponding to dynamic properties of the disturbances, or after each significant change of the disturbance values if these changes are irregular but can be measured or estimated on-line. After each solution of (1.21) the updated set-point values are transmitted to the feedback controllers.

In a case when explicit static process model (1.2) is available, the optimization problem (1.21) takes formally a simpler form

$$\begin{aligned} &\min Q(c,y) \\ &\text{subj. to: } y = F(c,w) \\ &\qquad\quad c \in C \\ &\qquad\quad y \in Y \end{aligned} \tag{1.22}$$

It may be not enough to take into account certain technological constraints on process output values in the formulation of the constrained optimization problem only. In cases of significant constraints determining quality parameters of products and being active in nominal conditions, it is common to enforce satisfaction of these constraints by dedicated constraint controllers, designing the constraint control (set-point control) layer. This method, though more costly, has two primary advantages:

- A constraint is tightly kept not only in steady-states, but also in periods of dynamic transients in the process, *e.g.*, after changes of disturbances.
- Applying the feedback loop eliminates (with an accuracy of the control error) unavoidable effects of the model mismatch and disturbance estimate errors used in the optimization problem. Therefore, the possibility of a violation of the constraints in the real process is practically eliminated – which would be not the case if the constraints were present in the optimization problem only.

Figure 1.3 in the previous section presents the control structure of this type, where forcing the equality constraint on a sub-vector $y^d$ of the vector $y$ of the process outputs,

$$y^d(t) = y^d_r$$

is implemented by application of a constraint controller. Let us observe that in this case the necessary information sent from the optimization unit to the subordinate controllers are optimal values of the set-points $c^f$ and $y^d_r$, although all elements of the vector $c = (c^f, c^d)$ are decision variables of the constrained optimization problem. However, transmission of the remaining part, $c^d$, to the constraint controller may also be needed, as only the whole information defines the new optimal steady-state process operating point. In particular, when using a predictive constraint controller the case $\dim c^d > \dim y^d$ may happen and be reasonable – then the predictive controller should know optimal steady-state value of $c^d$ to operate optimally (the case will be addressed in Chapter 4).

In the static optimization problem (1.21) or (1.22), there occurs the vector $w$ representing uncontrolled process input values (disturbances), in particular those which are significant for the process optimality. In classical multilayer structure it is usually assumed that the disturbances considered at the optimization layer are slow-varying, when compared to the controlled process dynamics. However, if variability of these disturbances is not so slow, in the worst case even comparable with the process dynamics, then values of the set-points should be updated more often. It is especially important in cases

when predictive controllers are applied, as then specialized efficient solutions to this problem are possible. These will be discussed in Chapter 4, which is devoted to the set-point optimization, after presenting model predictive control in Chapter 3.

From a point of view of the optimization, the disturbances $w$ are parameters. But the problem is that the model of the process $F$ should describe well the reality. Therefore, values of these disturbances should be known – measured or estimated. Only then the solution of the steady-state optimization problem with the process model does determine the operating point (set-point) which is optimal for the real process. More precisely, it is then close enough to the real optimal, but not strictly optimal due to unavoidable uncertainties and inaccuracies. If the case is not so, then the point evaluated using an only rough model can be far from the optimal one for the real process. In situations with significant uncertainty, but when the disturbances can be treated as constant during long time intervals, much longer than the controlled process settling time (*e.g.,* in cases of only roughly known abrupt but rare changes in disturbance values), there is an approach which allows to improve optimality of the operating point. It is based on an iterative use of an additional measurement information from the plant and results in the so-called *iterative set-point optimizing control algorithms* [16]. These algorithms will also be considered in Chapter 4.

**Example 1.2**
Let us go back to the previously considered CSTR reactor from Example 1.1. This time we shall formulate a different optimization objective (b):

(b) *To optimize the set-points on-line in such a way as to maximize concentration $C_B(t)$ in a product stream in the situation of a slowly changing disturbing input $F_A(t)$.*

The assumption of the inflow rate $F_A$ changing slow in relation to the dynamics of the controlled system allows us to use a steady-state control. For each disturbance value $F_A$ it is possible to determine, from (1.15), a temperature value which maximizes the concentration $C_B$. A set of optimal temperatures is marked by a dashed curve in Fig. 1.11, defined by points

$$\hat{T}(\frac{W_m}{F_A}) = \arg\max_T C_B(T, \frac{W_m}{F_A})$$

maximizing the function (1.15).

The task of the optimization layer is an on-line evaluation of a point of this curve corresponding to the current value of $F_A$. The optimizer can do this task by on-line optimization, performed each time. However, due to the simplicity of the task it is also possible to use a different method: the entire optimal curve $\hat{T}(\frac{W_m}{F_A})$ can be evaluated off-line by performing the optimizations for a sufficiently dense set of possible disturbance values and saved in the optimizer's memory, as a look-up table.

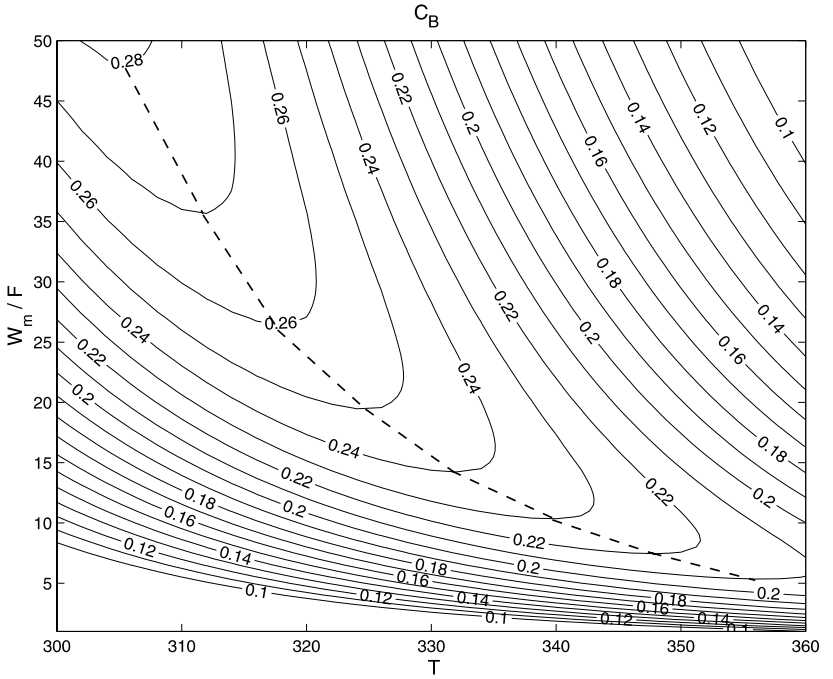


**Fig. 1.11.** Level sets of the surface $C_B(T, \frac{W_m}{F})$ with a dashed curve indicating the set of points maximizing $C_B$ (the set of optimal temperatures)

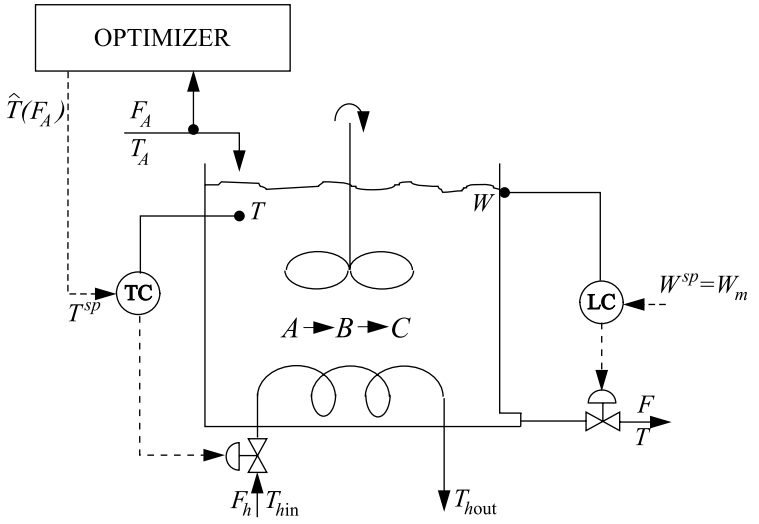


**Fig. 1.12.** Two-layer CSTR control structure with on-line set-point optimization

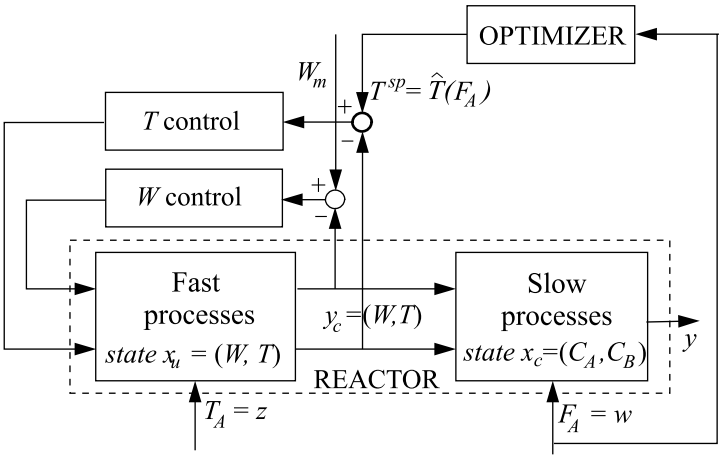


**Fig. 1.13.** Decomposed block diagram of the CSTR multilayer control system for on-line set-point optimization ($C_B$ maximization)

The control structure performing the control goal (b) is presented in Fig. 1.12, while Fig. 1.13 presents a diagram of the same control system in the language of multilayer structure blocks from Fig. 1.3, with a decomposed description of the process. □

## 1.6 Supervision, Diagnosis, Adaptation

In what have now become classic publications devoted to multilayer control of complex industrial processes, there were three basic layers distinguished, following from a functional decomposition of the control goal: a direct control (stabilization) layer, an optimization layer and an adaptation layer, see *e.g.*, [78, 40]. This philosophy followed the engineering experience up to the 1970s, when mechanical (pneumatic) and analog electronic single-loop feedback controllers were installed in industrial control systems. Computers, expensive and still not sufficiently reliable in those days, were being introduced to perform tasks of higher layers, mainly for acquisition, collecting and processing the data for purposes of operator and engineering staff support in their more or less automated decisions. The adaptation layer was responsible for basic supervision and adaptation activities in a control and decision support center. This center, on the basis of an analysis of a current process situation and external requirements, was making decisions about changes of the process operating points (set-points), about methods of enforcing quality requirements, about reaction to unpredictable events, *etc.* – which were then transmitted to the

functionally lower layers of direct control and optimization and implemented often manually by the operators.

The development of microelectronics and computer technology has led to radical changes in hardware and software of control systems. Due to an immensely developing miniaturization, along with the appearance of integrated circuits of a large and very large scale of integration, not only did the calculation possibilities rise immensely, but also prices and sizes of electronic and microprocessor equipment dropped. Therefore, along with the appearance of microprocessor controllers, and then distributed control systems, there came a new age in the technique of regulatory (feedback) control, in industrial processes control. In each of the functional and equipment layers of the control system, and even in a single direct control loop it is now possible to install more complex algorithms, including nonlinear algorithms and adaptive algorithms with automatic adaptation to assumed requirements.

Apart from the control algorithms, it is also possible to perform, in the same local controllers, the tasks of technical diagnosis of measurement signals and of operation of the controllers themselves, with an automatic shift to redundant units when needed. Therefore, the tasks of supervision and diagnosis dispersed significantly. One cannot talk of a single supervision, diagnosis and adaptation layer located on the top of the optimization layer. Along with the still existing central tasks of supervision and diagnosis and a task of adaptation of structure and parameters of the optimization layer, we also have tasks of supervision and adaptation of the direct or constraint control systems, which in large part are performed locally as local algorithms, more or less integrated with control algorithms and safety logic. Such control structures ensure greater speed of reaction to external events and an increased reliability – according to a general rule: information-decision loops should be made as short and quick as possible. A multilayer control structure considering up-to-date realizations of supervision, diagnosis and adaptation tasks is presented in Fig. 1.14 (compare with Fig. 1.2) [134, 16].

The problems of technical diagnosis are not considered in this book. The reader interested in this topic is referred to the excellent book [65].

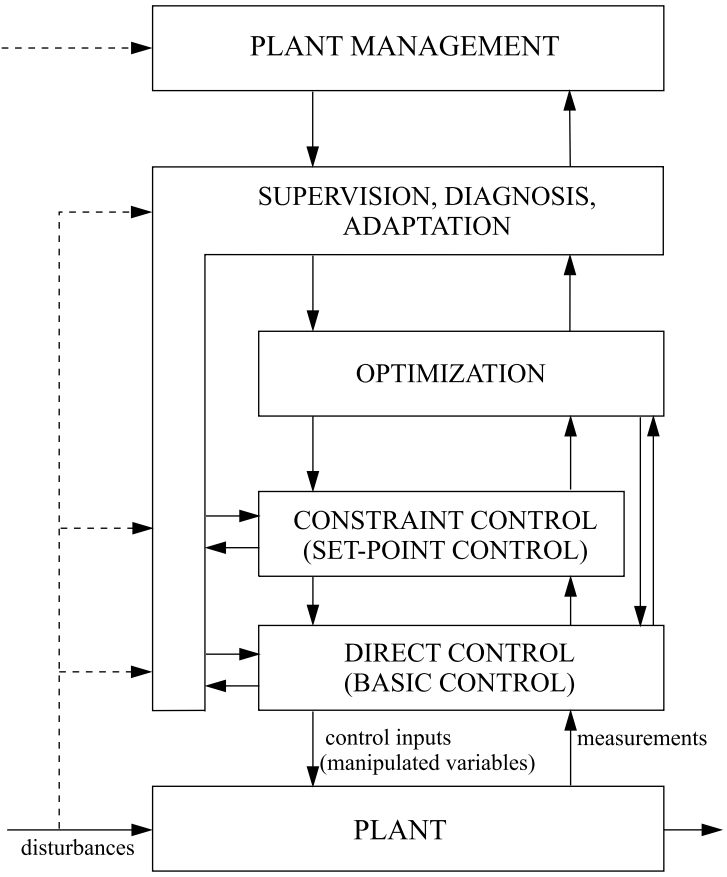


**Fig. 1.14.** Multilayer structure with supervision, diagnosis and adaptation tasks (reproduced with modifications from Brdys, M.A. and Tatjewski, P., *Iterative Algorithms for Multilayer Optimizing Control*, page 10, copyright 2005 by Imperial College Press, used by permission)

# 2

# Model-based Fuzzy Control

The task of feedback control systems is to keep controlled variables as close as possible to their reference values (set-points), which define operating points of the controlled process. A classical approach when selecting a feedback control algorithm is to first identify a linear process model at a given operating point, then to design a linear controller for that model or choose settings of a standard linear controller, usually of a PID type, see *e.g.*, [44, 85, 3, 52]. However, a linear process model is only an approximate description of reality. Moreover, the designed controllers should be robust against possible changes in the characteristics of the controlled process caused by disturbances or changes of its internal features. Robust controllers should ensure stability of the control system and basic control quality for a prescribed range of differences between the process and its model used for the design. For example, when applying classical frequency based design, certain minimal values of amplitude and phase margins must be preserved, assigned on the basis of frequency characteristics of the process model with the controller. These prescribed minimal values result from many years of practical experience, to achieve a reasonable compromise between robustness and control quality. A similar philosophy is applied when designing controllers by other methods, such as the root-locus method, pole placement, or popular in the process industries selection of PID settings using Ziegler-Nichols rules or tables of settings based on simple process models, see *e.g.*, [38, 44, 85, 105]. Such design usually ensures a correct operation of the control system in a certain neighborhood of an operating point for which the controller was designed. The smaller the nonlinearities of the process, the larger this neighborhood, in general.

Practically, however, control processes are usually nonlinear, sometimes strongly nonlinear. If they operate in a small neighborhood of the operating point and control quality requirements are not too high, then the presented design philosophy is usually sufficient. It had to be the case during many years when controllers were of mechanical construction (*e.g.*, pneumatic) as well as later, when analog electronic controllers dominated. Bringing into control practice computers and reliable microprocessor controllers radically changed

the situation. Supervisory control computers enabled effective implementation of on-line optimization of the controlled process, *i.e.*, proper adjustment of process operating points to the current technological situation defined by the process environment and requirements of the management layer. Moreover, microprocessor controllers allow for on-line realization of many complex control algorithms, including multivariable control and nonlinear control. It should be emphasized that the phenomena mentioned above condition each other strongly – on-line optimization of the running process assumes frequent, but always fast and reliable (thus - automatic) changes of the process operating points, implemented as changes in the set-point values for feedback controllers. This is possible only when controllers are able to work reliably not only in the vicinity of one set-point, but also for a range of set-points corresponding to various changes in process input and output values – *i.e.*, controllers which are able to control a nonlinear process.

Generally, the development of control algorithms capable to cope with changing operating points and environment changes has gone in two directions: adaptive control and nonlinear control. The base of *adaptive control* is an on-line adaptation of controller parameters to the changing process features, usually using a standard linear controller (*e.g.*, PID). Adaptation is conducted in a direct way or in an indirect way, by on-line identification of a linear process model corresponding to the current operating point and appropriate selection of controller parameters, usually using one of the classical methods as was described previously, see *e.g.*, [2, 103]. Such an approach is appropriate mainly in situations where we are not able to avoid the necessity of on-line identification during the control system operation. However, on-line identification carries the risk of a failure, particularly in periods of small variability of measured values. Therefore, in the domain of industrial control – in chemical, petrochemical, sugar, food *etc.* industries adaptive control in this sense has so far found quite a limited application.

The essence of *nonlinear control* (non-adaptive) is the design of a nonlinear controller using in a straightforward way a nonlinear process model, valid for a wide range of variability of process input and output values. The theory of nonlinear control is vast – see *e.g.*, [57, 64], and its description is not the aim of this book. This chapter will deal with the design and analysis of nonlinear controllers on the basis of the theory of fuzzy sets and fuzzy logic, particularly fuzzy models with the Takagi-Sugeno structure, proposed in 1985 in [131].

The idea, definitions of a *fuzzy set* and a *fuzzy logic* were proposed by L. A. Zadeh as early as in 1964 [156], at first meeting severe criticism. Although the first successful industrial application for control purposes took place in 1976 (at a Danish cement plant), the real boom in developing the theory and applications of fuzzy logic for control came at the end of the 1980s and lasts until today. It turned out that fuzzy controllers can be a strong, efficient tool, especially in places where it is difficult to have a sufficiently adequate pure analytical description of the controlled process – but where we do have at our disposal empirical knowledge, experience of operators controlling such proces-

ses, or patterns of the required behavior of the controlled processes. On the other hand, fuzzy control of nonlinear processes turned out to be a very efficient tool allowing to effectively combine elements of quality knowledge about the process with the analytical approach. Fuzzy modeling of nonlinear processes for control purposes turned out to be, next to neural network modeling, the most intensively developed approach which has been practically applied from the 1990s.

It is not the intention of this book to describe the basics of the theory of fuzzy sets and fuzzy logic together with possible applications for control purposes, as there exist many splendid books, such as *e.g.*, [155], where vast information on the subject can be found. The basics of the theory of fuzzy sets and fuzzy logic will be presented in the following section, however, only as necessary for introduction of the Takagi-Sugeno (TS) fuzzy modeling structure. We shall then present design methods and stability analysis of *fuzzy nonlinear controllers of Takagi-Sugeno structure*, also known as *fuzzy multi-regional controllers* [33, 32], or *multi-model controllers* [22]. It is not only the authors opinion that this is one of the most successful constructions of nonlinear controllers, especially from a practical, application point of view.

## 2.1 Takagi-Sugeno (TS) Type Fuzzy Systems

### 2.1.1 Fuzzy Sets and Linguistic Variables

Figure 2.1 (b) presents how membership of an element $x \in \mathbb{R}^1$ to a fuzzy set $F$ is defined. On the other hand, Fig. 2.1 (a) presents, for comparison, the membership of an element $x \in \mathbb{R}^1$ to the set $C$ defined in a classic way, such a set is called a *crisp set* in the theory of fuzzy sets. Each element of the numerical axis $\mathbb{R}^1$ belongs to the set $C$ or not, the membership function $\mu_C(x)$ of the set $C$ can take only values 0 or 1,

$$\mu_C(x) = \left\{ \begin{array}{ll} 1, & \text{if } x \in C \\ 0, & \text{if } x \notin C \end{array} \right\}$$

The membership function $\mu_F(x)$ of the fuzzy set $F$ can also take any value between 0 and 1,

$$\mu_F(x) = \left\{ \begin{array}{ll} \in (0,1], & \text{if } x \in F \\ 0, & \text{if } x \notin F \end{array} \right\}$$

Figure 2.1 presents an example of a trapezoidal membership function $\mu_F(x)$ of the fuzzy set $F$. Every point of the interval $[d, e]$ belongs only to the set $F$, *i.e.*, with the membership function value (with the *grade of membership*) equal to 1, just like each point of the interval $[a, b]$ belongs to the crisp set $C$. Points of intervals $(c, d)$ and $(e, f)$ do not belong entirely to the set $F$ because corresponding grades of membership are contained in the range $(0, 1)$. In this sense the borders of the set $F$ are *fuzzy*, thus the name *fuzzy set*. For example,

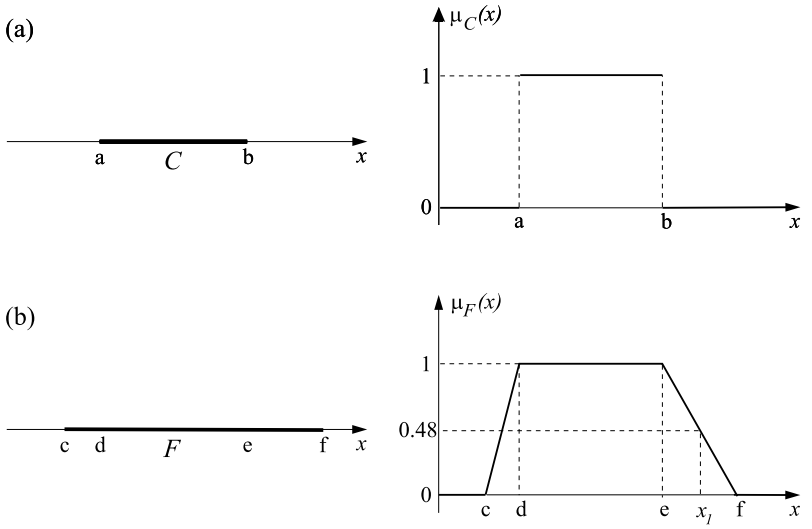


**Fig. 2.1.** (a) An example of a crisp set $C$, compared to (b) An example of a fuzzy set $F$; $\mu_C(x)$, $\mu_F(x)$ – membership functions of sets $C$ and $F$

the point $x_1$ depicted in Fig. 2.1 (b) belongs to the set $F$ with the membership function value (with the grade of membership) equal to 0.48.

The definition of a fuzzy set allows to introduce the next concept, key to the fuzzy logic, the concept of a *linguistic variable*. A linguistic variable, also known as a fuzzy variable, is an intermediate between a numerical variable and a symbolic variable (whose values are symbols – *e.g.*, a symbolic variable "shape" defined as taking three values: "circle", "square", "triangle"). The notion of the linguistic variable is explained in Fig. 2.2 by way of an example of a variable "temperature", taking values "low", "medium" and "high", defined by fuzzy sets described by membership functions $\mu_L(t)$, $\mu_M(t)$, and $\mu_H(t)$, respectively. For example, the temperature $t_1$ presented in Fig. 2.2 is medium with grade of membership equal to 0.79 and, at the same time, it is high with the grade of membership equal to 0.21.

Figures 2.1 and 2.2 show the application of fuzzy sets with *trapezoidal membership functions*. Together with *triangular functions* (which are special cases of trapezoidal functions, as in Fig. 2.1 with $d = e$) they are the most popular membership functions in applications where smoothness of these functions is not necessary. If, however, differentiability is required as *e.g.*, during the design of neuro-fuzzy systems (see Section 2.1.4), then the most popular in applications are sigmoidal functions, bell (bell-like) functions or Gaussian functions, see *e.g.*, [58, 155, 111].

The one-sided sigmoidal membership function (of a set $X_i$) is defined by the formula

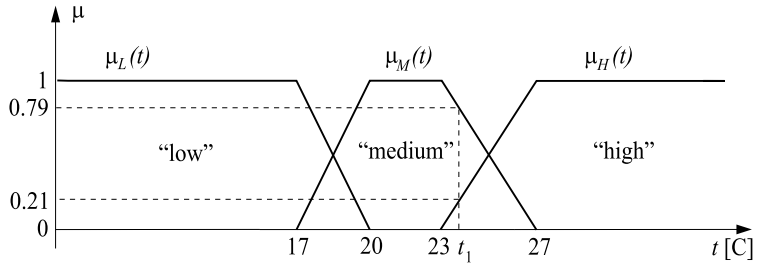


**Fig. 2.2.** Illustration of a linguistic variable "temperature" assuming three values: "low", "medium" and "high"

$$\mu_{X_i}(x) = \frac{1}{1 + \exp[-\alpha_i(x - c_i)]} \tag{2.1}$$

where parameters $\alpha_i$ and $c_i$ define the shape of a fuzzy set $X_i$. For $\alpha_i < 0$ the function is left-sided open, whereas for $\alpha_i > 0$ it is right-sided open. The value $c_i$ describes the positioning of the function ($\mu_{X_i}(c_i) = 0.5$), while $\alpha_i$ describes the steepness of its slope. One-sided sigmoidal functions can only define fuzzy sets corresponding to extreme values of linguistic variables. However, it is possible to build a two-sided (closed) sigmoidal membership function by the the use of two one-sided sigmoidal functions. Such a function, with its value approaching zero when $|x| \to \infty$, would represent intermediate values of a linguistic variable. The construction can be done in two simple ways: by subtracting two right-sided (or left-sided) open functions appropriately positioned in relation to each other, or by multiplying a right-sided open function by a left-sided open function positioned to its right (see *e.g.*, *Fuzzy Logic Toolbox* of the MATLAB® package). Figure 2.3 presents three sigmoidal membership functions:

- sigmf1: left-sided open with parameter values $\alpha_i = -30,\ c_i = 0.2$;
- sigmf2: two-sided created from subtracting function sigmf3 from a right-sided open function with parameter values $\alpha_i = 30,\ c_i = 0.2$;
- sigmf3: right-sided open with parameter values $\alpha_i = 15,\ c_i = 0.75$.

The popularity of sigmoidal functions results not only from the fact that it is easy to construct one-sided as well as two-sided sigmoidal functions, but also from the possibility to obtain asymmetry of two-sided functions – parameters of left and right slopes can be completely different and, at the same time, the size of the middle area (with a function value approximate to 1) can be shaped independently.

The generalized *bell membership function* is defined as follows [58, 155]

$$\mu_{X_i}(x) = \frac{1}{1 + [(\frac{x - c_i}{\alpha_i})^2]^{\beta_i}}, \tag{2.2}$$

where $\alpha_i, \beta_i, c_i$ are parameters defining the shape of a fuzzy set $X_i$. In particular, $c_i$ defines the centre of symmetry, $\beta_i$ influences mainly the inclination

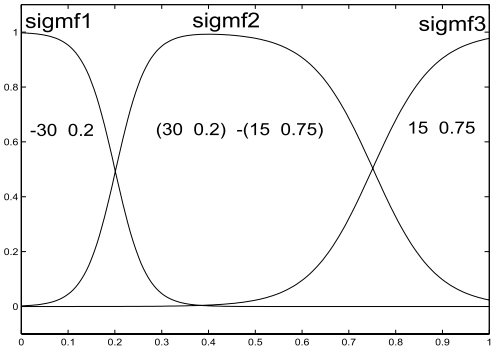


**Fig. 2.3.** Sigmoidal membership functions

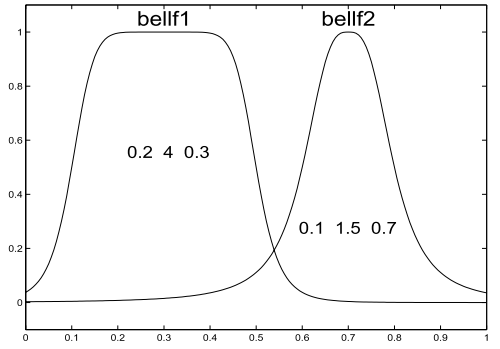


**Fig. 2.4.** Generalized bell membership functions

of slopes and $\alpha_i$ influences the width of the "bell". Figure 2.4 presents two examples of a bell function:

- bellf1: a function with parameter values $\alpha_i = 0.2,\ \beta_i = 4,\ c_i = 0.3$;
- bellf2: a function with parameter values $\alpha_i = 0.1,\ \beta_i = 1.5,\ c_i = 0.7$.

Let us note that the bell function is symmetric with respect to the vertical axis $x = c_i$. In order to obtain a two-sided asymmetric function, it should be built from two halves of bell functions with different parameters. In the literature and software packages Gaussian membership functions can be found as well, see *e.g.*, [58, 155, 111].

The linguistic variables (fuzzy variables) allow to capture roughly defined phenomena better than numerical variables, defined also as crisp variables. For example, the description of a linguistic variable "temperature", presented in Fig. 2.2 can result from a survey conducted among a representative sample of clients or communal users, *e.g.*, in order to establish conditions of turning

on heating appliances (when the temperature is low) or air-conditioning (when the temperature is high).

For a specific value of a numerical variable the process of defining its linguistic value together with the appropriate value of the membership function is called *fuzzyfication*. The example in Fig. 2.2 shows three linguistic values for the variable "temperature": "low", "medium" and "high", while in the process of fuzzyfication the linguistic values "medium" and "high", with the membership function values 0.79 and 0.21, respectively, are assigned to the fixed temperature numerical value $t_1$. Let us emphasize that a value of a linguistic variable is defined by a membership function assigned to this linguistic value, *i.e.*, it is identical with the fuzzy set defined by the corresponding membership function.

### 2.1.2 Fuzzy Reasoning

The basic element of a fuzzy system used *e.g.*, for modeling or control, is a *set of fuzzy inference rules* also known as a *knowledge base*, like in expert systems. It consists of inference rules operating on fuzzy (linguistic) variables, thus they are described as *fuzzy rules*.

Each inference rule consists of two elements: the IF-part, called an *antecedent* of a rule, and the THEN-part, also called a *consequent* of the rule. The structure of a single rule can thus be presented as follows:

$$\text{IF } <\text{antecedent}> \text{ THEN } <\text{consequent}>$$

The antecedent defines the condition, and the consequent – the conclusion which will be implemented if the condition is true.

The antecedent of a fuzzy rule consists, in the simplest case, of a single condition. Then the rule takes the following form:

$$\text{IF } x \text{ is } A \text{ THEN } <\text{consequent}>$$

where $x$ is a linguistic variable, while $A$ is a fuzzy set defined by a membership function $\mu_A(x)$. In the general case the rule antecedent can contain many simple conditions connected by the operators of conjunction (and), disjunction (or) and negation (not), *e.g.*,

$$\text{IF } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } (\text{not} A_2) \cdots \text{ or } x_k \text{ is } A_k \cdots \text{ THEN } <\text{consequent}>$$

The consequent of a fuzzy inference rule can, generally, take one of the following forms (*e.g.*, [155]):

1. Crisp consequent:

$$\text{IF } <\text{antecedent} > \text{ THEN } y = y_a$$

where $y_a$ is a numerical value or a symbolic value,

2. Fuzzy consequent:

$$\text{IF } <\text{antecedent}> \text{ THEN } y \text{ is } Y_k$$

where $y$ is a fuzzy variable (linguistic) and $Y_k$ is a fuzzy set,

3. Functional consequent:

$$\text{IF } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_2 \cdots \text{ and } x_n \text{ is } A_n \text{ THEN } y = f(x_1, x_2, \ldots, x_n)$$

where $f$ is a certain function of variables $x_1, x_2, \ldots, x_n$.

In this book we shall be interested only in functional consequents proposed in 1985 by Takagi and Sugeno [131]. It turned out that using the inference rules with functional consequents enables effective modeling of nonlinear dependencies using a small number of rules. Fuzzy systems which use rules with functional consequents are called, from the names of the authors, the *Takagi-Sugeno fuzzy systems* or simply *TS fuzzy systems*. A term also sometimes used is that of Takagi-Sugeno-Kang (TSK) systems, as Kang was one of Professor Sugeno's younger co-workers developing fuzzy systems of that structure. One may also come across a term *multiple model systems*, see *e.g.*, [22]. Let us note that the functional consequent is reduced to a crisp one (numerical) when the function $f$ assumes an extreme form of a constant, $f(x_1, x_2, \ldots, x_n) = a_0$. The most frequently applied functional consequents are those in the form of first order polynomials (affine functions)

$$f(x_1, x_2, \ldots, x_n) = a_0 + \sum_{i=1}^{n} a_i x_i$$

where $a_0, a_1, \ldots, a_n$ are parameters (polynomial coefficients). There are two reasons for this: the first and most important one is that using only affine functions is sufficient for a satisfactory precise modeling of even strongly nonlinear dependencies and with not a very large number of fuzzy sets in rule antecedents, provided a correctly selected number and location of these sets is chosen. Secondly, there exists a number of simple, well-established methods of construction and identification of linear models and of design of linear controllers which can be directly applied in individual sub-regions separated by antecedents of fuzzy TS system rules. Therefore, in this chapter we shall only use linear or affine functions of rule consequents, although the presented modeling and control structures can also be applied in a more general case with nonlinear functions.

Let us remember that a set of all inference rules of a fuzzy system, *e.g.*, of a fuzzy model of a nonlinear process, is called the *knowledge base* of that system. We shall assume that the knowledge base of a TS fuzzy system consists of rules in the following form

$$\mathrm{R}^i\text{: IF } x_1 \text{ is } A_1^i \text{ and } x_2 \text{ is } A_2^i \cdots \text{ and } x_n \text{ is } A_n^i$$
$$\text{THEN } y^i = a_0^i + \sum_{j=1}^{n} a_j^i x_j \qquad (2.3)$$

$i = 1, ..., r$. Each fuzzy set $A_j^i$ in the antecedent of a rule $\mathrm{R}^i$ is one of the elements of a set of linguistic values (fuzzy sets) of a variable $x_j$

$$A_j^i \in \mathbb{X}_j = \{X_{j1}, X_{j2}, ..., X_{jr_j}\} \qquad (2.4)$$

where $r_j$ is a number of elements of the set $\mathbb{X}_j$, $j = 1, ..., n$. The presented structure of the knowledge base of the TS fuzzy system will be illustrated by way of an example.

**Example 2.1**
Let us consider modeling of a two-dimensional nonlinear dependence $y = f(x_1, x_2)$ by a TS fuzzy system, using a two-element representation of each of the linguistic variables $x_1$ and $x_2$

$$x_1 \in \{\text{“small”, “big”}\} = \{X_{1m},\ X_{1d}\}$$
$$x_2 \in \{\text{“small”, “big”}\} = \{X_{2m},\ X_{2d}\}$$

The knowledge base will be the set of four rules $\mathrm{R}^i$, $i = 1, 2, 3, 4$:

$$\mathrm{R}^1 : \text{IF } x_1 \text{ is } X_{1m} \text{ and } x_2 \text{ is } X_{2m} \text{ THEN } y = y^1 = a_0^1 + a_1^1 x_1 + a_2^1 x_2$$
$$\mathrm{R}^2 : \text{IF } x_1 \text{ is } X_{1m} \text{ and } x_2 \text{ is } X_{2d} \text{ THEN } y = y^2 = a_0^2 + a_1^2 x_1 + a_2^2 x_2$$
$$\mathrm{R}^3 : \text{IF } x_1 \text{ is } X_{1d} \text{ and } x_2 \text{ is } X_{2m} \text{ THEN } y = y^3 = a_0^3 + a_1^3 x_1 + a_2^3 x_2$$
$$\mathrm{R}^4 : \text{IF } x_1 \text{ is } X_{1d} \text{ and } x_2 \text{ is } X_{2d} \text{ THEN } y = y^4 = a_0^4 + a_1^4 x_1 + a_2^4 x_2$$

where $X_{1m}$, $X_{2m}$, and $X_{1d}$, $X_{2d}$ are fuzzy sets defining “small” and “large” values of the linguistic variables $x_1$ i $x_2$, while the functional consequents are linear approximations of the modeled dependence over appropriate two-dimensional fuzzy sets. By presenting the formulated above four general rules in the form (2.3), we get

$$A_1^1 = X_{1m}, \quad A_2^1 = X_{2m}$$
$$A_1^2 = X_{1m}, \quad A_2^2 = X_{2d}$$
$$A_1^3 = X_{1d}, \quad A_2^3 = X_{2m}$$
$$A_1^4 = X_{1d}, \quad A_2^4 = X_{2d}$$

where the sets $A_1^i$, $i = 1, ..., 4$, are elements of the set (2.4) taking the form

$$\mathbb{X}_1 = \{X_{11},\ X_{12}\} = \{X_{1m},\ X_{1d}\}$$

where $r_1 = 2$, while $A_2^i$, $i = 1, ..., 4$, are elements of the set (2.4) taking the form

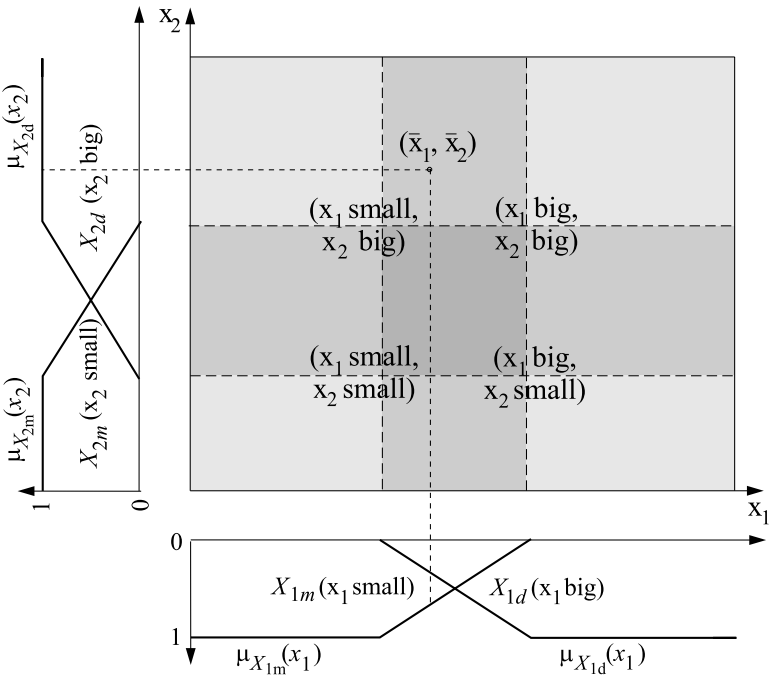


**Fig. 2.5.** A division of a two-dimensional domain in the case of two-element linguistic variables $x$ and $y$

$$\mathbb{X}_2 = \{X_{21},\ X_{22}\} = \{X_{2m},\ X_{2d}\}$$

where $r_2 = 2$.

An example of the discussed domain of the modeled dependence is presented in Fig. 2.5. The point $(\bar{x}_1, \bar{x}_2)$ marked in Fig. 2.5 belongs, with non-zero values of the membership functions, to the sets $X_{1m} \times X_{2d}$ ($x_1$ small, $x_2$ big) and $X_{1d} \times X_{2d}$ ($x_1$ big, $x_2$ big). Particularly, the coordinate $\bar{x}_1$ belongs to the set $X_{1m}$ with the membership function value $\mu_{X_{1m}}(\bar{x}_1) = 0.62$ and to the set $X_{1d}$ with the membership function value $\mu_{X_{1d}}(\bar{x}_1) = 0.38$, while the coordinate $\bar{x}_2$ belongs only to set $X_{2d}$, with the membership function value $\mu_{X_{2d}}(\bar{x}_2) = 1.0$.

The presented example illustrates not only a set of rules, but it also shows how naturally Cartesian products of fuzzy sets are created. □

A set of rules is used for the fuzzy reasoning. Generally, the fuzzy reasoning consists of three basic stages and, for some applications, additionally of a fourth stage (see *e.g.*, [155]):

1. Calculation of *levels (degrees) of activation* of all rules, also described as *firing strengths* of the rules, see *e.g.*, [59], corresponding to current numerical values of input variables.

2. Evaluation of *conclusions* of individual rules.
3. Combining the conclusions of all rules into one *final conclusion.*
   In the general case the (output) variable of a rule consequent is a fuzzy variable, thus the final conclusion is fuzzy, *i.e.*, $y \in Y$, where $y$ is an output variable of a fuzzy system and $Y$ is a fuzzy set created as a result of stages 1, 2 and 3 of the fuzzy reasoning. For certain applications, *e.g.*, for control, the obtained fuzzy value of the output variable should be further transformed into a crisp numerical form – then the following is performed:
4. *Defuzzification* – transforming a fuzzy value of the output variable into a numerical value.

The fuzzy reasoning, and in particular its third stage, is much more simplified when consequents of all rules are not fuzzy, if they are crisp or functional. This situation occurs in the case of TS fuzzy systems, which are the subject of our interest in this book. Therefore, we shall restrict our attention to this case only – referring a reader interested in more general aspects of fuzzy reasoning to the vast literature on the subject, *e.g.*, [58, 155].

**Fuzzy Reasoning in TS Structures**

In the case of TS fuzzy structures fuzzy reasoning can be divided into the following three stages:

1. Calculation of *activation levels* of individual inference rules corresponding to the current numerical values of the input variables.
2. Calculation of values of *functional consequents* of all individual inference rules.
3. Calculation of the *final conclusion* value – weighted and normalized sum of values of the output variables obtained in the rule consequences by considering the activation levels of the individual rules.

In the considered case of the TS fuzzy system each inference rule $\mathrm{R}^i$ has the form (2.3), $i = 1, ..., r$, where $r$ is the number of rules. Calculating the level of activation $w^i$ of the $i$-th rule, even in the considered case of the antecedent containing only simple conditions all connected by the conjunction operator, is *not a unique operation.* The multiplication operator or the minimum operator are frequently used here. When using the *multiplication operator* for the input variable values $x = [x_1\ x_2\ \ldots\ x_n]^T$ the level of activation of the $i$-th rule is given by the *algebraic product*

$$w^i(x) = \mu_{A_1^i}(x) \cdot \mu_{A_2^i}(x) \cdot \ \cdots \ \cdot \mu_{A_n^i}(x), \quad i = 1, ...r$$

while with the application of the *minimum operator* the level of activation is given by the *logical product*

$$w^i(x) = \min\{\mu_{A_1^i}(x), \mu_{A_2^i}(x), \ldots, \mu_{A_n^i}(x)\}, \quad i = 1, ...r$$

In the two-dimensional Example 2.1 presented in the previous section, for the value $x = \bar{x} = [\bar{x}_1\ \bar{x}_2]^T$ of the vector of the input variables (see Fig. 2.5) we obtain the following values of activation levels of individual rules, when using the minimum operator:

$$
\begin{aligned}
w^1(\bar{x}_1, \bar{x}_2) &= \min\{\mu_{X_{1m}}(\bar{x}_1) = 0.62,\ \mu_{X_{2m}}(\bar{x}_2) = 0\} = 0 \\
w^2(\bar{x}_1, \bar{x}_2) &= \min\{\mu_{X_{1m}}(\bar{x}_1) = 0.62,\ \mu_{X_{2d}}(\bar{x}_2) = 1\} = 0.62 \\
w^3(\bar{x}_1, \bar{x}_2) &= \min\{\mu_{X_{1d}}(\bar{x}_1) = 0.38,\ \mu_{X_{2m}}(\bar{x}_2) = 0\} = 0 \\
w^4(\bar{x}_1, \bar{x}_2) &= \min\{\mu_{X_{1d}}(\bar{x}_1) = 0.38,\ \mu_{X_{2d}}(\bar{x}_2) = 1\} = 0.38
\end{aligned}
$$

Further in this example, the following values of the output variable will be the values of the consequents of individual rules:

$$
\begin{aligned}
\mathrm{R}^1: &\quad \bar{y}^1 = a_0^1 + a_1^1\bar{x}_1 + a_2^1\bar{x}_2 \\
\mathrm{R}^2: &\quad \bar{y}^2 = a_0^2 + a_1^2\bar{x}_1 + a_2^2\bar{x}_2 \\
\mathrm{R}^3: &\quad \bar{y}^3 = a_0^3 + a_1^3\bar{x}_1 + a_2^3\bar{x}_2 \\
\mathrm{R}^4: &\quad \bar{y}^4 = a_0^4 + a_1^4\bar{x}_1 + a_2^4\bar{x}_2
\end{aligned}
$$

Let us note that using the product operator in the above example leads to the same result, which is a rather special case. Non-zero levels of activation calculated by minimum and product operators result in different values if at least two grades of membership occurring in a given rule are smaller than 1. But, directly from definitions of both operators, it is clear that zero levels of activation always occur for the same rules.

The last stage of fuzzy reasoning is a weighted, normalized sum of values of the output variables of consequents of all individual rules. For $r$ rules described by (2.3) this is obtained according to the formula

$$
y = \frac{\sum_{i=1}^{r} w^i(x) \cdot y^i}{\sum_{l=1}^{r} w^l(x)} = \frac{\sum_{i=1}^{r} w^i(x)[a_0^i + \sum_{j=1}^{n} a_j^i x_j]}{\sum_{l=1}^{r} w^l(x)} \tag{2.5}
$$

where $y^i$ are values of the consequents of individual rules at a point $x$, and $y$ is the final conclusion – the value of the output variable of a TS fuzzy system at the point $x$. For the value $\bar{x}$ of the input variables in Example 2.1 considered above we have

$$
\begin{aligned}
\bar{y} &= \frac{\sum_{i=1}^{4} w^i(\bar{x}) \cdot \bar{y}^i}{\sum_{l=1}^{4} w^l(\bar{x})} = \\
&= 0.62 \cdot (a_0^2 + a_1^2\bar{x}_1 + a_2^2\bar{x}_2) + 0.38 \cdot (a_0^4 + a_1^4\bar{x}_1 + a_2^4\bar{x}_2)
\end{aligned}
$$

Let us note that in the above example $\sum_{l=1}^{r} w^l(x) = 1$ for each value of $x$, which results from the construction of the membership functions. Generally, this condition does not have to be satisfied, thus there is a normalizing sum in

the fraction denominator in the formula (2.5), consisting of values of activation levels of all rules. Values

$$\tilde{w}^i(x) = \frac{w^i(x)}{\sum_{l=1}^{r} w^l(x)}, \quad i = 1, ..., r \tag{2.6}$$

are called *normalized activation levels* of the inference rules of a fuzzy model. They always satisfy the condition $\sum_{i=1}^{r} \tilde{w}^i(x) = 1$.

From the point of view of the relations between the input variables $x_1$, $x_2$, ..., $x_n$ and the output variable $y$, the TS fuzzy system can be treated as a functional mapping with features dependent on properties of the membership functions and rule consequent functions. If the membership functions are differentiable, as it is in the case of sigmoidal functions, the functions of the rule consequents are differentiable (they are usually affine) and, additionally, if levels of activation are defined by the product operator, then the nonlinear mapping generated by the TS fuzzy system will also be a continuous and differentiable mapping. In the two-dimensional case it is convenient to present the mapping of a TS fuzzy system in graphic form, as a surface $y = \varphi(x_1, x_2)$.

**Example 2.2**
Figure 2.6 presents the surface of a TS fuzzy system with two input variables $x_1$ and $x_2$, with fuzzy sets described by sigmoidal membership functions presented in Figures 2.7 and 2.8 and the following rule base of the general structure (2.3):

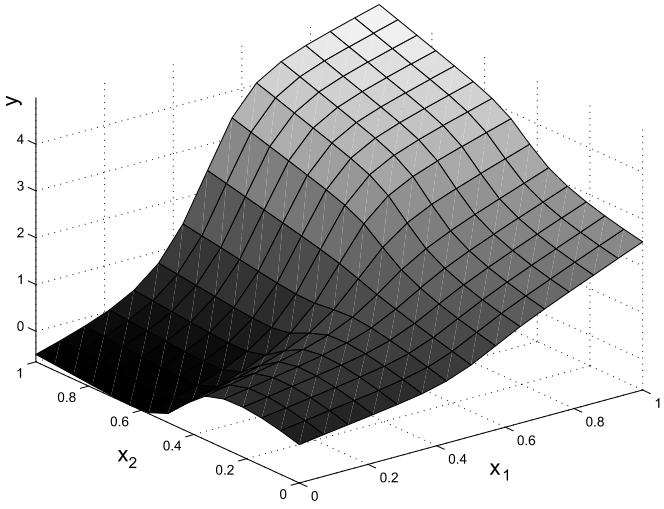


**Fig. 2.6.** Surface of the TS fuzzy system in Example 2.2

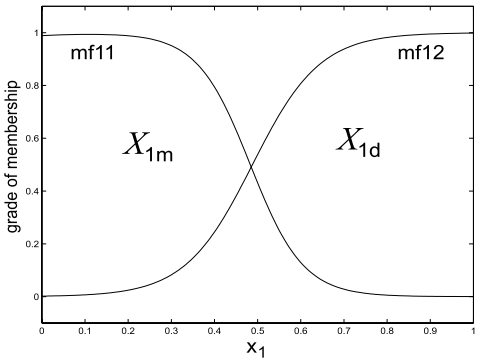


**Fig. 2.7.** Membership functions of fuzzy sets of the variable $x_1$

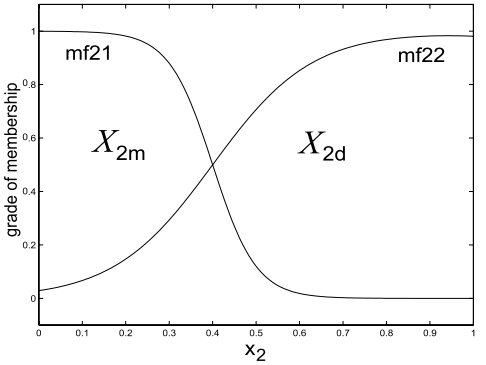


**Fig. 2.8.** Membership functions of fuzzy sets of the variable $x_2$

$$\begin{aligned}
\mathrm{R}^1: &\ \text{IF } x_1 \text{ is } X_{1m} \text{ and } x_2 \text{ is } X_{2m} \ \text{ THEN } y = 0.2 + x_1 + 2x_2 \\
\mathrm{R}^2: &\ \text{IF } x_1 \text{ is } X_{1m} \text{ and } x_2 \text{ is } X_{2d} \ \text{ THEN } y = -1 + 2x_1 + 0.5x_2 \\
\mathrm{R}^3: &\ \text{IF } x_1 \text{ is } X_{1d} \text{ and } x_2 \text{ is } X_{2m} \ \text{ THEN } y = 1 + 2x_1 + 2x_2 \\
\mathrm{R}^4: &\ \text{IF } x_1 \text{ is } X_{1d} \text{ and } x_2 \text{ is } X_{2d} \ \text{ THEN } y = -0.5 + 3x_1 + 1x_2
\end{aligned}$$

Let us note that in spite of simple affine consequents the obtained mapping is strongly nonlinear. □

### 2.1.3 Design of TS Fuzzy Models

Design of a TS fuzzy system, especially a fuzzy model of a known or unknown dependence between input and output variables, should be performed as follows:

1. *Fuzzy partitioning*: Dividing the range of variability of each input variable $x_j$ into partly overlapping sets $X_{jk}$, $k = 1, ... r_j$, see (2.4), *i.e.*, defining the number of fuzzy sets and assigning a shape and values of parameters of the membership function of each set.
2. Defining the structure and parameters of functional consequents of individual rules. The number of rules $r$ should be equal to the number of created sub-domains (multidimensional fuzzy subsets $A_1^i \times A_2^i \times \cdots \times A_n^i$, for the rules (2.3)) of the considered domain $X \in \mathbb{R}^n$ of input variables $x_j$, $j = 1, ..., n$, – one TS rule for each sub-domain.
3. Selecting the method of calculation of activation levels of individual rules: choosing the product or the minimum operator.

The first two points are the most important, the third one is a simple technical operation. A *proper selection of fuzzy sets* for input variables is a key to success. Too small a number of these sets and wrong positioning in relation to each other leads to unsatisfactory design of the fuzzy model, which does not satisfy the quality requirements and is too imprecise. Assuming too large a number of fuzzy sets leads to an oversized model with too many parameters; the design is then more difficult and the model is slower in operation.

There exist many proposals on how to perform the process of fuzzy partitioning in an automatic way, *e.g.*, using a data base consisting of a sufficiently representative set of values of input variables and corresponding output variables representing the modeled dependence. However, in practice it is still most efficient to take a human-made decision about the number of sets and their initial positioning, in an interactive mode, if necessary. This is precisely the place to employ empirical knowledge about the problem, about the character of nonlinearity. Human involvement is here not a drawback of the approach, on the contrary – it is its strong point, as the role of a human expert is well defined. Using expert knowledge is necessary to define a very important specific thing – proper general structure of a fuzzy system. The key is not to precisely define each membership function, but to define the number of these functions and their initial, approximate positioning. Precise tuning of parameters of membership functions is usually done automatically by using an appropriate optimization algorithm during a later phase of the design.

The way structure and parameters of functional consequents are defined depends on an application. Usually, polynomial functions of first order are employed. First of all, this is usually sufficient. Secondly, this leads to the possibility of simple and profitable interpretation of the fuzzy system as a neural network – as it will be presented later on. In a case of a fuzzy modeling the consequent functions will be affine approximations of the modeled dependence in individual sub-domains, *e.g.*, linearizations of this dependence at properly selected characteristic points of these sub-domains. In a case of the design of a fuzzy system as a controller, the functional consequents will be functions representing local control algorithms, designed for local models of the controlled process or directly for sets of data characterizing the process.

The selection of the type of the operator (minimum or product) for calculation of the activation levels of the rules is not very significant for the operation of a TS fuzzy system – it is arbitrary and dictated by the needs of the design method. The product operator is differentiable (the minimum operator is not), therefore it should be used if the nonlinear mapping represented by a TS fuzzy system should be differentiable. Especially, this situation occurs when membership function parameters are tuned in a learning mode of a fuzzy neural network.

### 2.1.4 TS System as a Fuzzy Neural Network

A TS fuzzy system can be presented in the form of a neural network structure, called a *fuzzy neural network* (FNN), or an ANFIS system (Adaptive Neuro-Fuzzy Inference System), see *e.g.*, [59, 58, 155]. Such a network can be interpreted as a multilayer perceptron in which nonlinear neuron nodes correspond to nonlinear membership functions. An example of the structure of a fuzzy neural network will be presented for a specific simple TS fuzzy system, where the domain of each of the two input variables $x_1$ and $x_2$ is divided into three fuzzy sets ($X_{11},\ X_{12}, X_{13}$ and $X_{21},\ X_{22},\ X_{23}$, respectively) defined by three sigmoidal membership functions: left-sided, two-sided and right-sided,

$$\mu_{X_{j1}}(x_j) = \frac{1}{1+\exp[-\alpha_{j1}(x_j - c_{j1})]},\ \ \alpha_{j1} < 0$$
$$\mu_{X_{j2}}(x_j) = \frac{1}{1+\exp[-\alpha_{j2+}(x_j - c_{j2+})]} - \frac{1}{1+\exp[-\alpha_{j2-}(x_j - c_{j2-})]}$$
$$\mu_{X_{j3}}(x_j) = \frac{1}{1+\exp[-\alpha_{j3}(x_j - c_{j3})]},\ \ \alpha_{j3} > 0$$

where $\alpha_{j2+} > 0,\ \alpha_{j2-} > 0,\ c_{j1} < c_{j2+} < c_{j2-} < c_{j3},\ \ j = 1, 2$. Let us consider a rule base consisting of three rules with antecedents containing only two simple conditions,

$$\mathrm{R}^1: \text{IF}\ \ x_1 \text{ is } X_{11}\ \ \text{and}\ \ x_2 \text{ is } X_{21}\ \ \text{THEN}\ \ y^1 = f_1(x) = a_0^1 + a_1^1 x_1 + a_2^1 x_2$$
$$\mathrm{R}^2: \text{IF}\ \ x_1 \text{ is } X_{12}\ \ \text{and}\ \ x_2 \text{ is } X_{23}\ \ \text{THEN}\ \ y^2 = f_2(x) = a_0^2 + a_1^2 x_1 + a_2^2 x_2$$
$$\mathrm{R}^3: \text{IF}\ \ x_1 \text{ is } X_{13}\ \ \text{and}\ \ x_2 \text{ is } X_{22}\ \ \text{THEN}\ \ y^3 = f_3(x) = a_0^3 + a_1^3 x_1 + a_2^3 x_2$$

where $x = [x_1\ \ x_2]^T$. We are considering three rules only for simplicity of presentation, a full base would contain 9 rules. The multiplication (product) operator is used for assigning levels of activation

$$w^1(x) = \mu_{X_{11}}(x) \cdot \mu_{X_{21}}(x)$$
$$w^2(x) = \mu_{X_{12}}(x) \cdot \mu_{X_{23}}(x)$$
$$w^3(x) = \mu_{X_{13}}(x) \cdot \mu_{X_{22}}(x)$$

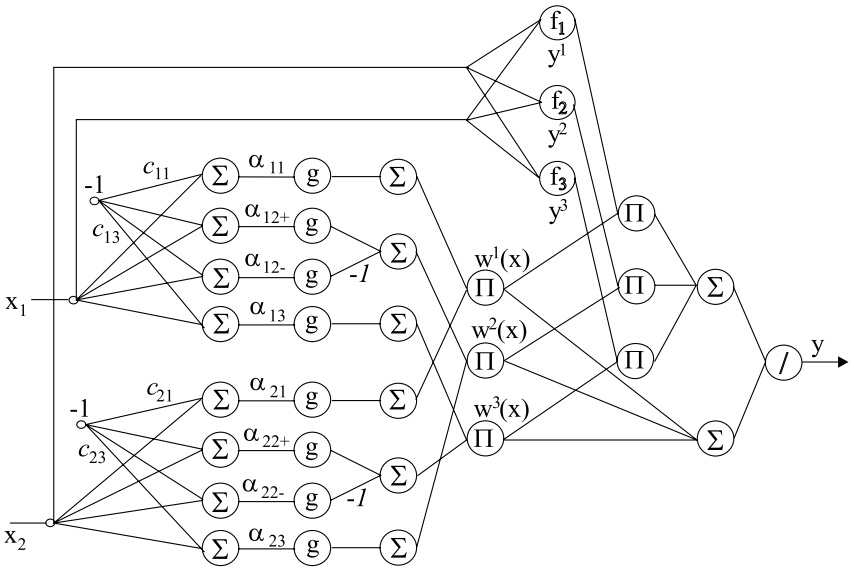


**Fig. 2.9.** Presentation of a TS fuzzy system, with two input variables and three rules, in the form of a fuzzy neural network (the case of constant parameters of functions $f_i$ in the consequents)

while the system output is given according to the standard formula

$$y = \frac{\sum_{i=1}^{3} w^i(x) \cdot y^i}{\sum_{l=1}^{3} w^l(x)} = \sum_{i=1}^{3} \tilde{w}^i(x) \cdot y^i \tag{2.7}$$

where $\tilde{w}^i(x)$ denote normalized activation levels of the rules.

The structure of the described fuzzy neural network (FNN), for the case of constant (*i.e.*, not adjustable by the network) parameters of functions in the rule consequents, is presented in Fig. 2.9. The network consists of eight layers (not including the trivial layer consisting of input nodes):

- The first four layers are connected with the rule antecedents and they are responsible for calculation of activation levels $w^i(x)$ of individual rules. The first three layers calculate values of the membership functions, where nodes of the second layer model nonlinear functions of the type

$$g(z) = \frac{1}{1 + \exp[-z]}$$

  while nodes of the fourth layer, denoted by "$\Pi$", are the multiplication nodes.

- Each node of the fifth layer calculates the value of the function $f_i(x)$ of one rule consequent; these calculations can be assigned to one node for each rule when coefficients of the functions are assumed to be constant (not tunable) network parameters.
- The sixth, seventh and eighth layers implement the final conclusion, according to (2.7).

In Fig. 2.9 the following *convention* is used: parameters $c_{ji}$ and $\alpha_{ji}$ are placed near the arches, as close to arch centers as possible – they are tuneable network parameters, by which signals leaving the nodes are multiplied (in case of $c_{ji}$ parameters, $c_{j2+}$ and $c_{j2-}$ were not shown due to lack of space, whereas $c_{j3}$ were placed closer to the beginning of the arches, $j = 1, 2$). Network arches which do not have any assigned values should be treated as arches with the value "1". Concerning nodes transforming signals, the markings by these nodes denote node output signals.

A fuzzy neural network can be used to adjustment (optimization) of parameters $\alpha_{ji}$ and $c_{ji}$ of the membership functions, by employing one of the standard network learning algorithms, see *e.g.*, [58, 155]. Certainly, it is possible in a situation when a set of input-output type learning data is available, in our example problem the data describing the dependence $x \rightarrow y$. By performing TS fuzzy modeling of a known nonlinear dependence one can generate such a set. However, it is a typical situation when a TS fuzzy model of an unknown dependence is constructed, based only on a set of input-output data. After the user has selected the number of fuzzy sets, initial shapes of membership functions and structure of rule consequents, it is possible to tune parameters of the membership functions as well as parameters of the consequent functions using methods of network learning. There are two methods available:

- We can extend the network replacing the function nodes "$f_i$" by additional network fragments presented in Fig. 2.10 – and then tune parameters of

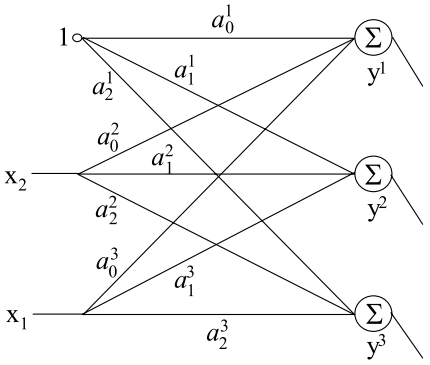


**Fig. 2.10.** Fragment of the network structure implementing functions of rule consequents

the membership functions and of the rule consequents by a single, selected method of network learning, *e.g.*, using a gradient optimization with gradient calculation according to the back propagation algorithm.

- A *hybrid optimization algorithm* can be used (see [58]): the parameters of the rule antecedents are tuned by a method of network learning (a gradient optimization using the back propagation algorithm to evaluate the gradient), alternately with tuning the parameters of the rule consequents by the least squares method. The authors of [58] have conducted comparative research which indicated better properties of the hybrid algorithm.

**Example 2.3**

The example will present results of fuzzy modeling of static characteristics of a distillation column shown in Fig. 2.11, used for separation of a two-component mixture of methanol and water. A dynamic model of the column was constructed [67], in the form of a set of differential and algebraic equations describing physical phenomena occurring on individual shelves of the column, in the reflux tank and in the bottom part of the column [81, 112, 46]. Models of controllers stabilizing liquid levels of the top and bottom products by manipulating outflows of the distillate $D$ and of the bottom product $B$ were then added.

Assuming all time derivatives in the model equations constant and equal to zero, a set of nonlinear algebraic equations was obtained, which describes static characteristics of the column. Input values (process control inputs) of

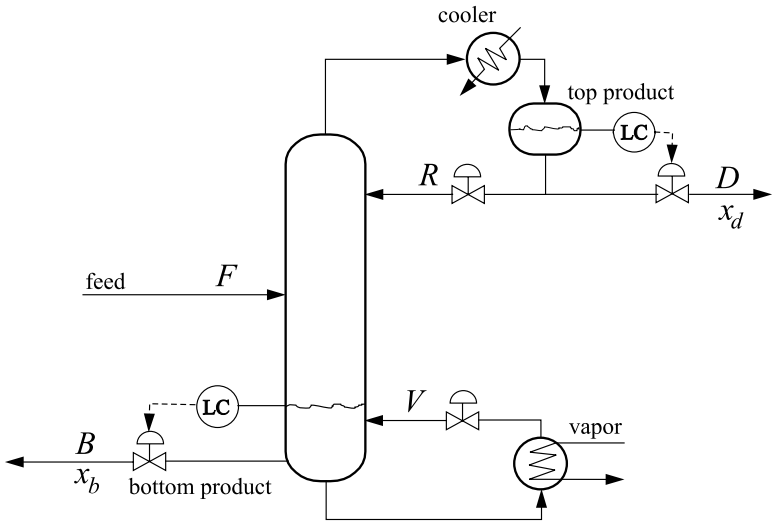


**Fig. 2.11.** Distillation column, Example 2.3

these characteristics are: the reflux (a stream of recycle from the reflux tank to the column) flow rate $R$ and the heating steam flow rate $V$ $[kmol/h]$. The outputs are concentrations $x_d$ and $x_b$ of the product (methanol) in the distillate $D$ and in the bottom product $B$, respectively. Figure 2.12 presents original, nonlinear surfaces of the static characteristics of the column.

Starting to build a TS fuzzy model, a division of each of the domains of the input variables $R$ and $V$ into two partitions (two fuzzy sets) was assumed and sigmoidal membership functions were chosen (arbitrarily), as presented in Fig. 2.13 (a) and Fig. 2.14 (a). In this way, four fuzzy sub-domains (four two-dimensional fuzzy sets) were created in the domain of each of the characteristics, compare with Fig. 2.5 and with Example 2.1. In each of these sub-domains affine models for concentrations $x_d$ and $x_b$ were assumed,

$$x_d = a^i_{d0} + a^i_{d1}R + a^i_{d2}V, \quad i = 1, ..., 4$$
$$x_b = a^i_{b0} + a^i_{b1}R + a^i_{b2}V, \quad i = 1, ..., 4$$

For identification of parameters $a^i_{dj}$ and $a^i_{bj}$ of local models a set of data was created for modeling purposes: in 441 (21 x 21) evenly distributed points covering the entire range of variability of $R$ and $V$ the values of static characteristics (these shown in Fig. 2.12) were calculated. Next, parameters of the local models were tuned to this data using the method of minimization of the mean square deviation. The surfaces of TS fuzzy models obtained in this way are presented in Fig. 2.15.

Next, the model parameters were optimized using the hybrid method, performing alternately:

- steps modifying the parameters of the membership functions, in the direction of a negative gradient (calculated by the back propagation method)

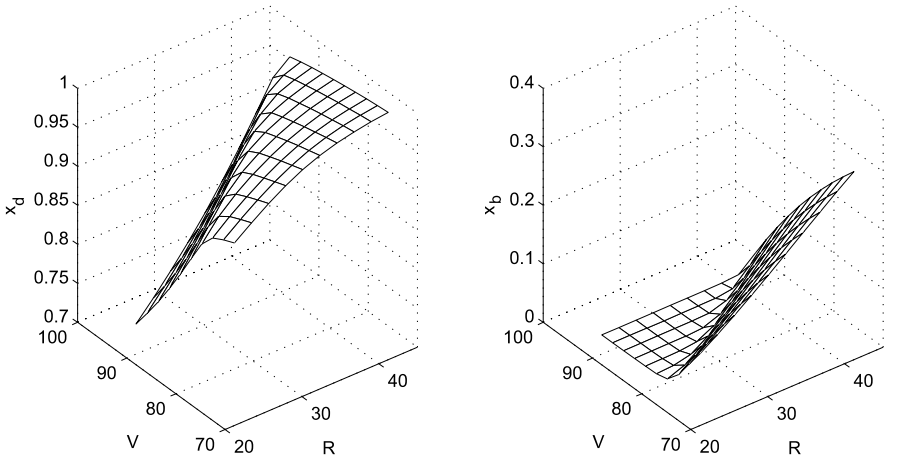


**Fig. 2.12.** Surfaces of the static characteristics of the plant, Example 2.3

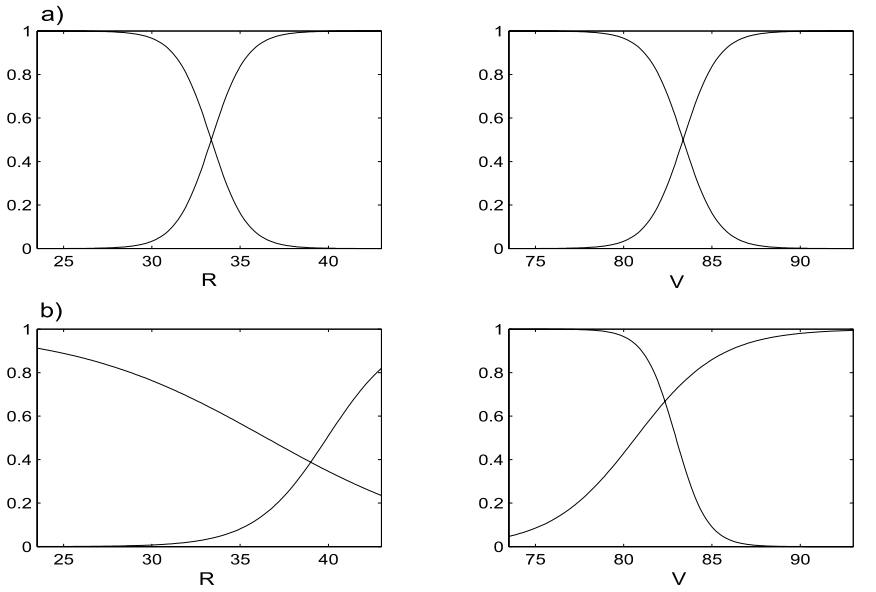


**Fig. 2.13.** Membership functions of fuzzy sets of the TS model of concentration $x_d$: a) initial, b) after parametric optimization

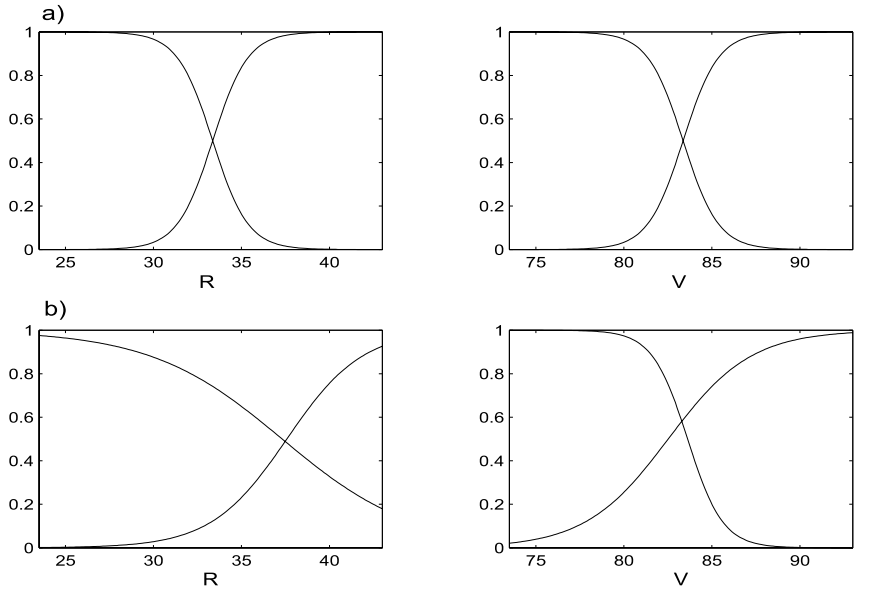


**Fig. 2.14.** Membership functions of fuzzy sets of the TS model of concentration $x_b$: a) initial, b) after parametric optimization

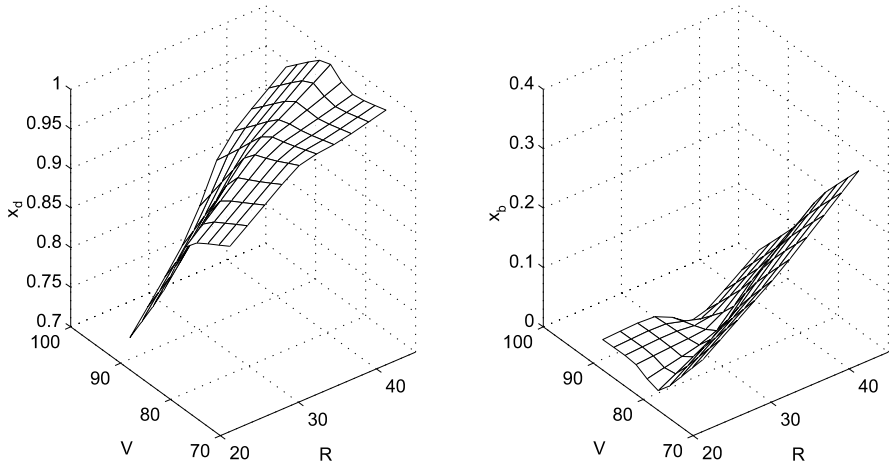


**Fig. 2.15.** Surfaces of TS fuzzy models of static characteristics of concentrations $x_d$ and $x_b$, for initial values of parameters of membership functions

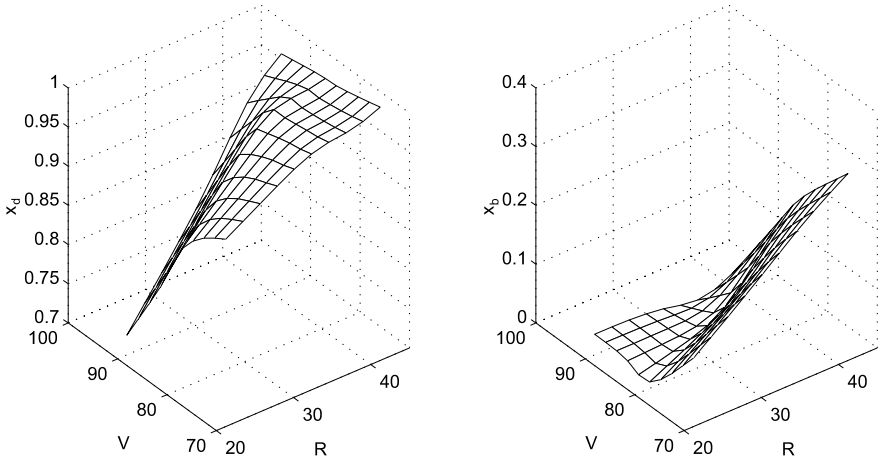


**Fig. 2.16.** Surfaces of TS fuzzy models of static characteristics of concentrations $x_d$ and $x_b$, for optimal values of parameters of membership functions

of the performance function representing the mean square modeling error and

- steps tuning the parameters of the affine models (functions of the rule consequents), by the least squares method.

There were 500 iterations performed, after that the value of the modeling error was $8.45\times10^{-3}$ for the model of $x_d$ and $7.49\times10^{-3}$ for the model of $x_b$.

Shapes of the membership functions obtained in this way are presented in Fig. 2.13 (b) and Fig. 2.14 (b), while surfaces of the fuzzy models are shown in Fig. 2.16. What draws our attention is a high modeling accuracy of the strongly nonlinear mapping obtained using a small number of fuzzy sets. □

## 2.2 Discrete-time TS Fuzzy Control

By a *TS fuzzy control algorithm* we shall describe a TS fuzzy system in which the consequent of each rule is a function defining a control algorithm, designed to control the process locally in a sub-domain (multidimensional local fuzzy set) associated with the rule antecedent. The presented approach is also called in the literature a *parallel distributed compensation* (PDC), see *e.g.*, [150], a *fuzzy multi-regional control algorithm* [33, 32] or a *multi-model control algorithm* [22].

Control algorithms of rule consequents of TS fuzzy controllers are generally designed locally, for local models describing the process in particular sub-domains. Generally, the local process models can be linear or nonlinear, as can the local control algorithms. In practice, linear local models and linear local control algorithms are applied, as they are sufficient for a nonlinear description if only the sub-domains are properly selected in relation to the nonlinearity of the process. When using a locally linear approach the design is very effective; it is possible to use known and practically well verified simple control algorithms. Thus, in this book, we shall restrict our attention to affine or linear local models and control algorithms. The structure of the design of rules of the TS fuzzy control algorithm in the case of a process described by a TS fuzzy model is presented in Fig. 2.17.

Design of a TS fuzzy controller can be divided into the following stages:

1. a) Define variables occurring in the rule antecedents. Values of these variables will describe the membership of the current state (operating point) of the process to *local sub-domains*, in which the process can be approximated by affine models, for control purposes.
   b) Divide the range of variability of each variable $x_j$ of the rule antecedents into overlapping fuzzy sets $X_{jk}$ (in relation to the process nonlinearity), defining the number $r_j$ (see (2.4)) of these sets and assigning a shape and values of parameters of the membership function of each set. This creates a division of the whole domain into a number of overlapping sub-domains (multidimensional fuzzy sets $A_1^i \times A_2^i \times \cdots \times A_n^i$, for the rules (2.3)).
   c) Formulate an affine or linear process model for each sub-domain constructed so far (each sub-domain is defined by one fuzzy rule, see Example 2.2).

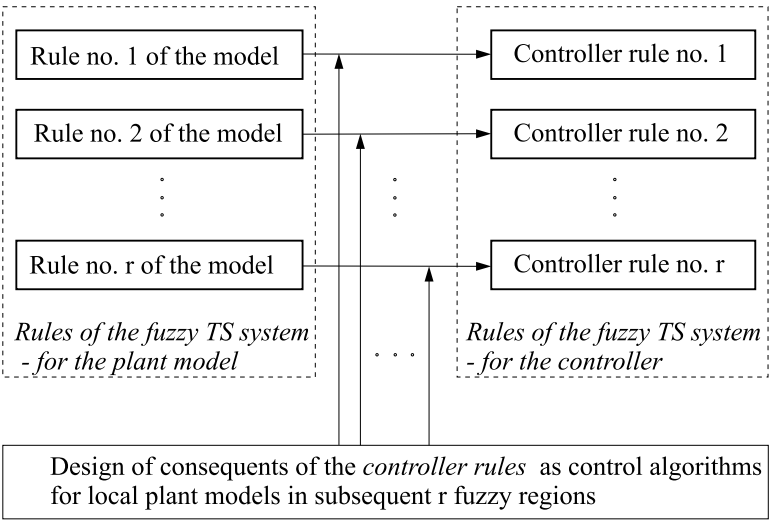


**Fig. 2.17.** Design of a TS fuzzy controller for a process modeled by a TS fuzzy system

The procedure defined above means a design of a TS fuzzy model of the considered nonlinear control process, with functional consequents which are linear dynamic models of the process in particular local sub-domains. The range of variability of the input and output signals should cover the entire operation domain of the designed controller.

2. For each local affine model (sub-model) of the process *design a linear controller* operating correctly in the assigned sub-domain, using a selected technique appropriate to the applied process model, *e.g.*, a state-feedback controller for models in the form of state equations, a PID controller for ARX models, *etc.* Create a TS fuzzy controller – a TS fuzzy system with rule antecedents created in the previous point (division into fuzzy sets as in the TS fuzzy model of the process) and functional consequents which are the designed control algorithms.

3. Perform *analysis* of the obtained TS fuzzy algorithm using *simulation and theoretical analysis*: if it is possible, check analytically whether stability conditions are satisfied and simulate the operation of the closed-loop control system in predicted conditions. If results are not satisfactory, return to the previous point and correct parameters of local controllers. Repeat the simulation and theoretical analysis of the obtained control system. If there are difficulties in obtaining the desired quality of control by means of tuning the local controllers only, return to the first stage: modify the fuzzy sets of the rule antecedents, *i.e.*, the parameters of the membership

functions, or even the number of partitions (number of fuzzy sets) into which the ranges of individual variables occurring in the rule antecedents were divided. Repeat the following design and analysis of the corrected control system.

It should be noted that the last point is significant. Properties of a TS fuzzy controller are not a simple collection of properties of local controllers. There are examples known, see *e.g.*, [133], when in spite of the stability of all local controllers, the TS fuzzy nonlinear controller is unstable.

Increasing the reliability and power of computer systems together with strong competition on the market of industrial products unavoidably lead to the application of multilayer control with on-line optimization of the operating points. Thus, in many loops of the direct control layer it is necessary to consider applications of nonlinear controllers, because in the situation of current, automatic changes in values of the set-points and significant nonlinearities, the frequently occurring PID linear controllers do not ensure adequate control quality. In such cases they have to be tuned for the worst case, *i.e.*, for operating points critical for stability and robustness – thus they operate too slowly in less critical conditions. The TS fuzzy controllers, discussed in this chapter, are appropriate not only for the set-point control layer (where sampling periods are usually larger) but they can be widely used for the direct feedback control. Moreover, if functions in the rule consequents of a TS fuzzy controller are linear, *e.g.*, describing the classical PID control algorithm, then this controller is a *natural nonlinear generalization of a linear PID controller*, a generalization which should be easily understood and accepted by the staff operating the control system.

The time domain is natural for the TS fuzzy systems used for process modeling or for constructing dynamic systems such as controllers, because conditions occurring in the rule antecedents of a process model or of a controller are formulated for variables in the time domain. Therefore, in functional consequents of fuzzy rules there occur affine or linear dynamic models which define values of output variables of a process or a controller at a given moment. A standard, practical approach when selecting settings of an industrial PID controller is the use of a largely simplified process model, mainly in the form of a simple transfer function, obtained experimentally on the basis of an output response to a step function or to a rectangular impulse – and reading the appropriate settings from predefined tables, see *e.g.*, [38, 44]. There is nothing wrong with the application of this method for the design of a fuzzy PID controller, only local process models corresponding to individual local fuzzy sets (sub-domains) should be presented in the form of appropriate transfer functions and used for selection of the local PID settings.

The process of tuning the parameters of the local controllers, or even jointly parameters of the controllers and the membership functions, can also be done globally and in an automatic way – if we have a credible simulation model of the controlled process at our disposal. Then, we can use optimization

of a function formulated as an appropriate control quality index, usually also with constraints, *e.g.*, on the overshoot. On the other hand, if we have a set of data representing desired behavior of the control system, then by presenting the designed TS fuzzy controller in the form of a fuzzy neural network, it is possible to apply an algorithm for optimization (teaching) of such a network (see Section 2.1.4).

The discussion presented above concerns general problems of the design of nonlinear TS fuzzy controllers. In the current state of technology these controllers are implemented in microcomputers, so they are discrete (digital) controllers. However, if the sampling period is small enough in relation to the dominant process time constant, then a fully authorized method of the design is the synthesis of a continuous control algorithm – and then application of its digital representation only, see *e.g.*, [44, 52]. For that reason, after presenting design of discrete-time TS fuzzy controllers in the following sections, we shall discuss the same problem for the continuous-time case.

### 2.2.1 Discrete TS Fuzzy State-feedback Controllers

The first case of a TS fuzzy controller discussed in the literature was a state-feedback controller [133, 150], based on a TS fuzzy model with consequents in the form of linear state equations. It was only later that output-feedback controllers were proposed, based on input-output type models.

Let us consider a TS fuzzy discrete model of a dynamic process with the rules of the following form:

$$\begin{aligned} \mathrm{R}_p^i: \quad \text{IF} \quad & x_1(k) \text{ is } A_1^i \text{ and } x_2(k) \text{ is } A_2^i \text{ and } \cdots \text{ and } x_{n_x}(k) \text{ is } A_{n_x}^i \\ & \text{THEN} \quad x^i(k+1) = \mathbf{A}_i x(k) + \mathbf{B}_i u(k) \end{aligned} \tag{2.8}$$

where

$$\begin{aligned} x(k) &= [x_1(k)\ x_2(k)\ \cdots\ x_{n_x}(k)]^T \\ u(k) &= [u_1(k)\ u_2(k)\ \cdots\ u_{n_u}(k)]^T \end{aligned}$$

are state and control input vectors of a dynamic process at sampling instant $k$, respectively, $A_j^i \in \mathbb{X}_j = \{X_{j1}, ..., X_{jr_j}\}$ are fuzzy sets of the coordinate $x_j$ of the state vector $x$ (see (2.4)), $j = 1, \ldots, n_x$, while $\mathbf{A}_i$ and $\mathbf{B}_i$ are the state and control matrices of local linear models created for local fuzzy sets (sub-domains), $i = 1, ..., r$.

Using a *grid partition* of the domain of the input variables [59] (see the partition presented in Fig. 2.5 as an example), we obtain a fuzzy system with the number of sub-domains (and rules) equal to

$$r = r_1 \cdot r_2 \cdot \ \cdots \ \cdot r_{n_x} = \prod_{j=1}^{n_x} r_j$$

As a result of this, the number of local fuzzy sets grows rapidly with the increase of dimensionality $n_x$ of the state vector $x$ and the number $r_j$ of fuzzy sets corresponding to each coordinate $x_j$ of the state vector. Thus, it is important that the domains of individual state variables are divided into as small a number of fuzzy sets as possible, *i.e.*, that the numbers $r_j$ are small. This is possible, without losing modeling accuracy, by using the TS structures - as opposed to classic fuzzy modeling, in which the rule consequents are fuzzy sets.

For given values of the state and control input vectors, $x(k)$ and $u(k)$, the output of a fuzzy model, *i.e.*, the state at the next sampling instant, is calculated according to the general formula (2.5), *i.e.*,

$$x(k+1) = \frac{\sum_{i=1}^{r} w^i(k)[\mathbf{A}_i x(k) + \mathbf{B}_i u(k)]}{\sum_{l=1}^{r} w^l(k)} \tag{2.9}$$

where $w^i(k)$ are levels of activation of individual rules (2.8) at $k-$th sampling instant,

$$w^i(k) = \prod_{j=1}^{n_x} \mu_{A_j^i}(x_j(k))$$

We have taken above the natural assumption that always $\sum_{l=1}^{r} w^l(k) > 0$, *i.e.*, for each value of the state variables from their domain the model is well defined – at least one rule is activated. It is convenient to operate with normalized rule activation levels, see (2.6), which from definition satisfy the condition $\sum_{i=1}^{r} \tilde{w}^i(k) = 1$. Then the model of an autonomous process (with $u(k) \equiv 0$)) takes the form

$$x(k+1) = \sum_{i=1}^{r} \tilde{w}^i(k) \mathbf{A}_i x(k) \tag{2.10}$$

For such a dynamic model, using directly the Lyapunov theorem (see *e.g.*, [61, 148]), it is relatively easy to derive a sufficient condition of asymptotic stability [133].

**Theorem 2.1** *The equilibrium point of a dynamic system (2.10) described by the rules (2.8) with $u(k) = 0$ is globally asymptotically stable, if there exists a symmetric and positive definite matrix $\mathbf{P}$, such that for the state matrix $\mathbf{A}_i$ of each local model the following inequality is satisfied*

$$\mathbf{A}_i^T \mathbf{P} \mathbf{A}_i - \mathbf{P} < \mathbf{0}, \quad i = 1, 2, ..., r \tag{2.11}$$

*Proof.* Let us formulate a scalar function of the form

$$V(x(k)) = x^T(k) \mathbf{P} x(k) \tag{2.12}$$

where $\mathbf{P}$ is a symmetric positive definite matrix. The function (2.12) satisfies the following conditions:

$$\begin{aligned} V(0) &= 0 \\ V(x(k)) &> 0 \ \text{ for } \ x \neq 0 \\ V(x(k)) &\to \infty \ \text{ for } \ \|x(k)\| \to \infty \end{aligned}$$

Moreover, a change of its value along a trajectory of the dynamic system (2.10) is defined by the formula

$$\begin{aligned} \Delta V(x(k)) &= V(x(k+1)) - V(x(k)) \\ &= x^T(k+1)\mathbf{P}x(k+1) - x^T(k)\mathbf{P}x(k) \\ &= x^T(k)\left[\sum_{i=1}^{r}\tilde{w}^i(k)\mathbf{A}_i^T\mathbf{P}\sum_{j=1}^{r}\tilde{w}_j(k)\mathbf{A}_j - \mathbf{P}\right]x(k) \\ &= \sum_{i=1}^{r}\sum_{j=1}^{r}\tilde{w}^i(k)\tilde{w}^j(k)x^T(k)[\mathbf{A}_i^T\mathbf{P}\mathbf{A}_j - \mathbf{P}]x(k) \\ &= \sum_{i=1}^{r}(\tilde{w}^i(k))^2x^T(k)[\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i - \mathbf{P}]x(k) + \\ &+ \sum_{i=1}^{r}\sum_{j<i}^{r}\tilde{w}^i(k)\tilde{w}^j(k)x^T(k)[\mathbf{A}_i^T\mathbf{P}\mathbf{A}_j + \mathbf{A}_j^T\mathbf{P}\mathbf{A}_i - 2\mathbf{P}]x(k) \end{aligned}$$

We have

$$\begin{aligned} \mathbf{A}_i^T\mathbf{P}\mathbf{A}_j + \mathbf{A}_j^T\mathbf{P}\mathbf{A}_i - 2\mathbf{P} = &-(\mathbf{A}_i - \mathbf{A}_j)^T\mathbf{P}(\mathbf{A}_i - \mathbf{A}_j) \\ &+ [\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i - \mathbf{P}] + [\mathbf{A}_j^T\mathbf{P}\mathbf{A}_j - \mathbf{P}] \end{aligned}$$

and it follows from the positive definiteness of the matrix $\mathbf{P}$ that

$$-(\mathbf{A}_i - \mathbf{A}_j)^T\mathbf{P}(\mathbf{A}_i - \mathbf{A}_j) \leq \mathbf{0}$$

Moreover,

$$\tilde{w}^i(k) \geq 0 \quad \text{for every } i = 1, 2, ..., r$$

thus $\Delta V(x(k)) < 0$ for $x(k) \neq 0$ if only $\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i - \mathbf{P} < \mathbf{0}$ for every $i = 1, 2, ..., r$. Then the function $V(x(k))$ is a Lyapunov function of the nonlinear dynamic system, which concludes the proof. □

It should be emphasized that the above theorem requires that only one matrix $\mathbf{P}$ common for all local linear dynamic models $x^i(k+1) = \mathbf{A}_i x(k)$ satisfies the stability condition. To find a solution (or to prove it does not exist) of the system of linear matrix inequalities

$$\begin{aligned} \mathbf{A}_i^T\mathbf{P}\mathbf{A}_i - \mathbf{P} < 0, \quad i = 1, 2, ..., r \\ \mathbf{P} > 0, \ \ \mathbf{P} \text{ symmetric} \end{aligned} \tag{2.13}$$

is not difficult now; one can use procedures from the *LMI Toolbox* of the MATLAB® package (LMI - Linear Matrix Inequalities).

It is worth mentioning here that for nonsingular matrices $\mathbf{A}_i$, *a necessary condition for the existence of a matrix* $\mathbf{P}$ *satisfying the Theorem 2.1 is that each of the product matrices* $\mathbf{A}_i\mathbf{A}_j$, $i, j = 1, 2, ..., r$ *is a matrix of an asymptotically stable system, i.e.*, that the modulus of any of its eigenvalues is smaller than 1 [133]. Namely, it follows from the condition (2.11) that

$$\mathbf{P} - (\mathbf{A}_j^{-1})^T\mathbf{P}(\mathbf{A}_j)^{-1} < 0$$

Adding the above inequality and (2.11) results in

$$\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i - (\mathbf{A}_j^{-1})^T\mathbf{P}(\mathbf{A}_j)^{-1} < 0$$

which can be converted to the form

$$\mathbf{A}_j^T\mathbf{A}_i^T\mathbf{P}\mathbf{A}_i\mathbf{A}_j - \mathbf{P} < 0, \quad i = 1, 2, ..., r$$

It follows from the last inequality that the matrix $\mathbf{A}_i\mathbf{A}_j$ must be asymptotically stable. Thus, proving that one of the matrices $\mathbf{A}_i\mathbf{A}_j$ is not asymptotically stable determines non-existence of the matrix $\mathbf{P}$ satisfying the conditions of Theorem 2.1.

Recently, a version of the Theorem 2.1 with weakened conditions has been obtained [151], for dynamic systems with trajectories of limited transitions between different subsets of the whole operating space. To present these results, the whole process operating state-space must be additionally partitioned into subsets of *constant activation of fuzzy rules* [60, 151], which in the sequel will be called *constant activation cells*, or *activation cells*, for brevity. The activation cells will be denoted by $S_l$, $l = 1, ..., L$, where $L$ is the number of all cells. Within each cell activation level of every fuzzy rule is either zero (inactive rule) or positive (active rule - its activation level may be zero only on the cell boundary).

The constant activation cells may be divided into those where the process is in an *operating regime* and those corresponding to *interpolating regimes*. The cell is with an operating regime when $\tilde{w}^l(x(k)) = 1$ for a certain $l \in L$ and all other normalized activation levels are equal to zero. Thus, the system dynamics is then fully defined by a local linear model being the consequent of the fuzzy rule defining the fuzzy sub-domain containing this cell. On the other hand, activation cells with interpolating regimes are defined as those where $0 < \tilde{w}^l(x(k)) < 1$ for $x(k) \in \text{int}S_l$ (interior of $S_l$) for at least two $l \in L$ (at least two, because $\sum_{l \in L} \tilde{w}^l(x(k)) = 1$ for every sampling instant $k$). Notice that all activation cells are crisp and not overlapping sets, creating a partition of the overall process state-space which is closely related to the partition of this space into fuzzy sub-domains defined so far (see the beginning of Section 2.2), but different, containing more elements.

An example of a partition of the problem state-space into fuzzy sub-domains and into cells of constant activation of fuzzy rules is shown in Fig. 2.18, for a TS fuzzy model consisting of the following three rules

$$\begin{aligned}
\mathrm{R}_p^1: &\ \text{IF} \quad x_1(k) \text{ is } X_1 \quad \text{THEN} \quad x(k+1) = A_1 x(k) + B_1 u(k) \\
\mathrm{R}_p^2: &\ \text{IF} \quad x_1(k) \text{ is } X_2 \quad \text{THEN} \quad x(k+1) = A_2 x(k) + B_2 u(k) \\
\mathrm{R}_p^3: &\ \text{IF} \quad x_1(k) \text{ is } X_3 \quad \text{THEN} \quad x(k+1) = A_3 x(k) + B_3 u(k)
\end{aligned}$$

where $x = (x_1, x_2)$, $0 \le x_2 \le 3$, $X_i$ are fuzzy sets defined by membership functions $\mu_{X_i}(x_1)$, $i = 1, 2, 3$, and 3 two-dimensional fuzzy sub-domains are defined by Cartesian products $X_i \times [0, 3]$, $i = 1, 2, 3$.

Note that, to preserve strict mathematical correctness, the partitioning of the process state-space into activation cells is possible only for trapezoidal (or trapezoidal-like) membership functions, as sigmoidal, gaussian or generalized bell functions have positive activation levels over the whole universe of discourse.

Denoting, for each activation cell $S_l$, by $K(l)$ a set containing indexes of all fuzzy rules with nonzero activation level over $S_l$ (for a cell $S_l$ with operating regime its corresponding $K(l)$ contains a single element), we can describe the autonomous (control free) fuzzy system dynamics as follows

$$x(k+1) = \sum_{i \in K(l)} \tilde{w}^i(k)[\mathbf{A}_i x(k)], \quad x(k) \in S_l, \quad l \in L \tag{2.14}$$

where $0 < \tilde{w}^i(k) = \tilde{w}^i(x(k)) \le 1$ for each $i \in K(l)$ and $\sum_{i \in K(l)} \tilde{w}^i(k) = 1$.

Define also, for further use, a set $\Omega$ representing *all possible one-step system state transitions* between different constant activation cells $S_l$,

$$\Omega = \{(l, m): \ x(k) \in S_l, \ x(k+1) \in S_m, \ \text{for every } l, m \in L\} \tag{2.15}$$

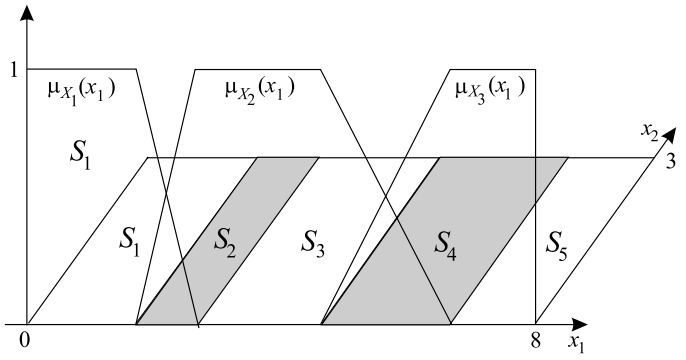


**Fig. 2.18.** Membership functions $\mu_{X_i}(x_1)$ defining 3 fuzzy sets $X_i$, 3 fuzzy sub-domains $X_i \times [0, 3]$ and corresponding 5 constant activation cells: 3 with operating regimes ($S_1$, $S_3$, $S_5$ – white) and 2 with interpolating regimes ($S_2$, $S_4$ – grey)

where $m \neq l$ when the system transits (in one step) from a cell $S_l$ to the cell $S_m$, whereas $m = l$ when the system stays in the same cell $S_l$. We are now in a position to formulate a version of Theorem 2.1 with weakened stability conditions, due to the use of a piecewise-quadratic Lyapunov function, instead of a quadratic one.

**Theorem 2.2** [151] *The equilibrium point of a dynamic system (2.14) described by the rules (2.8) with $u(k) \equiv 0$ is globally asymptotically stable, if there exist $L$ symmetric and positive definite matrices $\mathbf{P}_l$, $l \in L$ such that the following set of inequalities is satisfied*

$$\mathbf{A}_i^T \mathbf{P}_m \mathbf{A}_i - \mathbf{P}_l < 0, \quad \text{for every } (l,m) \in \Omega, \quad i \in K(l) \tag{2.16}$$

*Proof.* The reasoning is similar as in the proof of Theorem 2.1, only the system dynamics described by (2.14) and the following piecewise-quadratic Lyapunov function candidate

$$V(k) = x(k)^T \mathbf{P}_l x(k), \; x(k) \in S_l, \; l \in L \tag{2.17}$$

must be used, instead of the function (2.12). □

Theorems 2.1 and 2.2 state only sufficient conditions for stability – if these conditions are not satisfied then the question of stability remains open. According to the best knowledge of the author, necessary and sufficient stability conditions have yet not been formulated. Some attempts were made to obtain a formulation of sufficient stability conditions on the basis of considering properties of local state matrices $\mathbf{A}_i$ only, without the necessity of solving global or partitioned sets of inequalities, such as those in (2.11) or (2.16). In [20] conditions of this type were given, based on a direct estimate of the state norm of the system described by (2.10)

$$\begin{aligned} \|x(k+1)\| &= \|\sum_{i=1}^{r} \tilde{w}^i(k) \cdot \mathbf{A}_i x(k)\| \\ &\leq \sum_{i=1}^{r} \tilde{w}^i(k) \|\mathbf{A}_i\| \|x(k)\| \\ &\leq \max_{1 \leq i \leq r} \|\mathbf{A}_i\| \sum_{i=1}^{r} \tilde{w}^i(k) \|x(k)\| \\ &= \max_{1 \leq i \leq r} \|\mathbf{A}_i\| \cdot \|x(k)\| \end{aligned} \tag{2.18}$$

where $\|\mathbf{A}\|$ denotes a norm of the matrix $\mathbf{A}$, induced by the vector norm. Thus satisfaction of the condition

$$\sum_{i=1}^{r} \tilde{w}^i(k) \|\mathbf{A}_i\| < 1 \tag{2.19}$$

or the slightly stronger condition

$$\|\mathbf{A}_i\| < 1, \quad i = 1, 2, ..., r \tag{2.20}$$

ensures asymptotic stability of the dynamic system (2.10). The above conditions turn out to be very conservative in practice, and as such, of not much use.

The example given below, after [133], shows that stability of individual linear subsystems $x(k+1) = \mathbf{A}_i x(k)$ is generally not sufficient for the stability of the overall nonlinear fuzzy system. Thus, in order to ensure stability, additional conditions such as that in Theorem 2.1 or Theorem 2.2, are necessary.

**Example 2.4**
Let us consider fuzzy sets $X_1$ and $X_2$ presented in Fig. 2.19 and a TS fuzzy model consisting of the following two rules

$$\begin{aligned} \mathrm{R}_p^1: &\ \text{IF} \quad x(k-1) \text{ is } X_1 \quad \text{THEN} \quad x^1(k+1) = x(k) - 0.5x(k-1) \\ \mathrm{R}_p^2: &\ \text{IF} \quad x(k-1) \text{ is } X_2 \quad \text{THEN} \quad x^2(k+1) = -x(k) - 0.5x(k-1) \end{aligned}$$

which are equivalent to the rules (2.8) with state matrices

$$\mathbf{A}_1 = \begin{bmatrix} 1 & -0.5 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{A}_2 = \begin{bmatrix} -1 & -0.5 \\ 1 & 0 \end{bmatrix}$$

corresponding to the state definition

$$[\, x_1(k)\ x_2(k) \,] = [\, x(k)\ x(k-1) \,]$$

Eigenvalues of matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are $0.5 \pm 0.5i$ and $-0.5 \pm 0.5i$, respectively, thus the matrices describe asymptotically stable systems. But, starting the fuzzy system from an initial point $x_1 = 0.9,\ x_2 = -0.7$ we obtain a trajectory shown in Fig. 2.20, definitely indicating non-stability of the fuzzy nonlinear model. Assumptions of Theorem 2.1 can not be satisfied in this case. Indeed, it is easy to check that matrix $\mathbf{A}_1\mathbf{A}_2$,

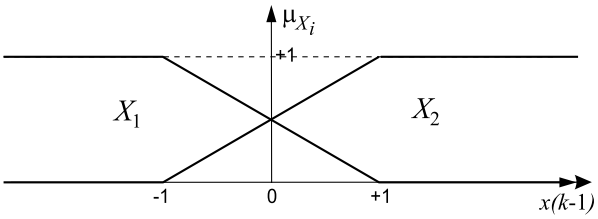


**Fig. 2.19.** Fuzzy sets of the model, Example 2.4

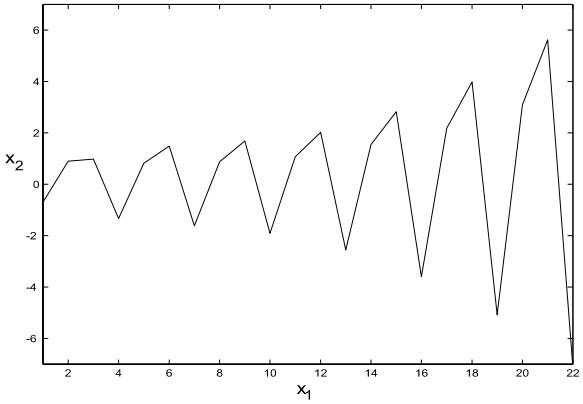


**Fig. 2.20.** Unstable state trajectory in Example 2.4

$$\mathbf{A}_1\mathbf{A}_2 = \begin{bmatrix} -1.5 & -0.5 \\ -1 & -0.5 \end{bmatrix}$$

has eigenvalues equal to $\lambda_1 = 0.134$, $\lambda_2 = 1.866$, thus it is not a matrix of a stable discrete dynamic system – therefore, the matrix $\mathbf{P}$ satisfying assumptions of Theorem 2.1 does not exist. □

For each local linear dynamic model from the consequents of the rules (2.8), that is for each of the sub-domains of a fuzzy model, a linear state-feedback controller can be designed using a standard method. In this way we obtain the rules of a TS fuzzy controller in the following form

$$\mathrm{R}_c^j: \quad \text{IF} \quad x_1(k) \text{ is } A_1^j \text{ and } x_2(k) \text{ is } A_2^j \text{ and} \cdots \text{ and } x_{n_x}(k) \text{ is } A_{n_x}^j \quad \text{THEN} \quad u^j(k) = -\mathbf{F}_j x(k) \tag{2.21}$$

where $\mathbf{F}_j$ are matrices of state-feedback coefficients, $j = 1, 2, ..., r$. A complete nonlinear TS fuzzy controller will be described by the following relation

$$u(k) = -\sum_{j=1}^{r} \tilde{w}^j(k)\mathbf{F}_j x(k) \tag{2.22}$$

The structure of a control system with such a controller is presented in Fig. 2.21.

When modeling and simulating the feedback control system we can, generally, use for process representation in the closed-loop a nonlinear process model different than the one used for the design of the fuzzy controller. For example, it can be a TS fuzzy model with a different number of partitions for certain state variables $x_j$ (with a different number and shape of fuzzy sets corresponding to those variables), and thus with a different number and parameters of the rule antecedents. If the number of these rules is denoted by $ro$,

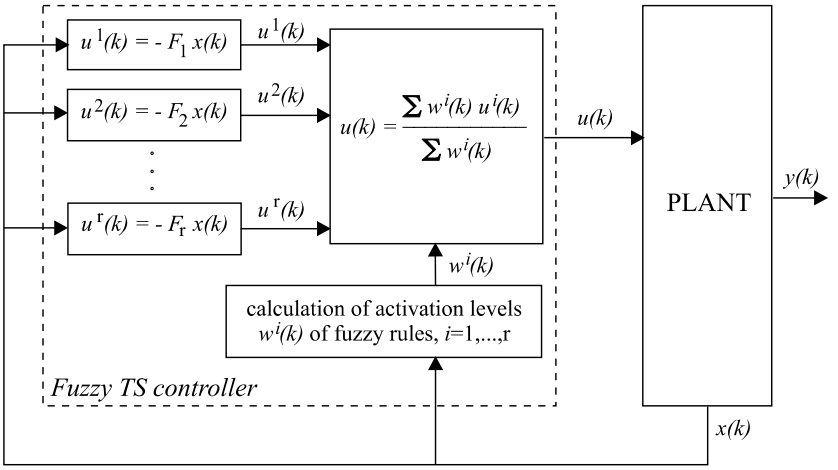


**Fig. 2.21.** Structure of a control system with the TS fuzzy state-feedback controller

and levels of their activation by $w_o^i(k)$, $i = 1, ..., ro$, then the model output will be given by the following formula, analogous to (2.9)

$$x(k+1) = \frac{\sum_{i=1}^{ro} w_o^i(k)[\mathbf{A}_i x(k) + \mathbf{B}_i u(k)]}{\sum_{l=1}^{ro} w_o^l(k)} = \sum_{i=1}^{ro} \tilde{w}_o^i(k)[\mathbf{A}_i x(k) + \mathbf{B}_i u(k)] \tag{2.23}$$

where $\tilde{w}_o^i(k)$ are normalized rule activation levels. Of course, if the TS fuzzy model used for process modeling in the feedback control system structure is different than the model (2.8) used for the design of the controller, then also its matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ are generally different (although we do not introduce different symbols here, as it does not lead to misunderstanding: when analyzing the closed-loop control system only matrices of the model (2.23) representing the process in the loop and controller matrices $\mathbf{F}_j$ occur in the control system description). Substituting the controller equations (2.22) into (2.23) we obtain the description of the closed-loop system in the following form

$$x(k+1) = \sum_{i=1}^{ro} \sum_{j=1}^{r} \tilde{w}_o^i(k) \tilde{w}^j(k)(\mathbf{A}_i - \mathbf{B}_i \mathbf{F}_j) x(k) \tag{2.24}$$

To examine the stability properties of the dynamic system (2.24) Theorem 2.1 can be directly used, however, only matrices $\mathbf{A}_i - \mathbf{B}_i \mathbf{F}_j$, $i = 1, 2, ..., ro$, $j = 1, 2, ..., r$, should be taken in place of matrices $\mathbf{A}_i$, $i = 1, 2, ..., r$ in its formulation.

However, if during modeling the closed-loop control system presented in Fig. 2.21 one uses the same TS fuzzy model for the plant description in the loop that was used for the fuzzy controller design (i.e. with the same number

of rules and the same rule antecedents), then the formula describing dynamics of the closed-loop control system will have the following form

$$x(k+1) = \sum_{i=1}^{r} \sum_{j=1}^{r} \tilde{w}^i(k)\tilde{w}^j(k)(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_j)x(k) \tag{2.25}$$

Then we can obtain a more convenient, simpler formulation of the stability conditions if we exploit the equality

$$\tilde{w}^i(k)\tilde{w}^j(k) = \tilde{w}^j(k)\tilde{w}^i(k), \quad i,j = 1,2,...,r$$

resulting from the identity of the rule antecedents in the process model and in the controller. Equation (2.25) can then be written in the form

$$x(k+1) = \sum_{i=1}^{r} \tilde{w}^i(k)\tilde{w}^i(k)(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_i)x(k) + 2 \sum_{i,j=1,i<j}^{r} \tilde{w}^i(k)\tilde{w}^j(k)\mathbf{D}_{ij}x(k) \tag{2.26}$$

where

$$\mathbf{D}_{ij} = \frac{1}{2}[(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_j) + (\mathbf{A}_j - \mathbf{B}_j\mathbf{F}_i)], \quad i < j,\ i,j = 1,2,...,r \tag{2.27}$$

while the symbol $\sum_{i,j=1,i<j}^{r}$ denotes addition of terms with all pairs of indexes $i,j$ such that $i < j$, $i,j = 1,2,...,r$, *e.g.*,

$$\sum_{i,j=1,i<j}^{3} \mathbf{D}_{ij} = \mathbf{D}_{12} + \mathbf{D}_{13} + \mathbf{D}_{23}.$$

Thus, directly from Theorem 2.1 we obtain

**Corollary 2.3** *Equilibrium point of the control system (2.25) described by the rules of the process model (2.8) and the controller (2.21) is globally asymptotically stable, if there exists a positive definite matrix* $\mathbf{P}$ *satisfying the following conditions*

$$(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_i)^T\mathbf{P}(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_i) - \mathbf{P} < 0,\ i = 1,...,r \tag{2.28}$$

$$\mathbf{D}_{ij}^T\mathbf{P}\mathbf{D}_{ij} - \mathbf{P} < 0,\ i < j,\ i,j = 1,...,r \tag{2.29}$$

*for all pairs* $(i,j)$ *except for those for which always (for every sampling instant* $k$*)* $w^i(k)w^j(k) = 0$. □

In [132] a certain weakening of the above result was achieved, the proof runs similarly as in Theorem 2.1, with the use of the same form of the Lyapunov function.

**Theorem 2.4** *Let the maximum number of rules activated at the same time in the control system (2.25) described by the rules of the process model (2.8)*

*and of the controller (2.21) be no higher than* $s$, $1 < s \leq r$. *The equilibrium point of this system is globally asymptotically stable if there exist a positive definite matrix* $\mathbf{P}$ *and a positive semi-definite matrix* $\mathbf{Q}$ *satisfying the conditions*

$$(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_i)^T\mathbf{P}(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_i) - \mathbf{P} + (s-1)\mathbf{Q} < 0, \quad i = 1, ..., r \tag{2.30}$$

$$\mathbf{D}_{ij}^T\mathbf{P}\mathbf{D}_{ij} - \mathbf{P} - \mathbf{Q} \leq 0 \quad i < j, \quad i, j = 1, ..., r \tag{2.31}$$

*for all pairs* $(i, j)$ *except for those for which* $w^i(k)w^j(k) = 0$ *for every sampling instant* $k$. □

Let us note that conditions (2.30) and (2.31) reduce to (2.28) and (2.29), if $\mathbf{Q} = \mathbf{0}$ is assumed.

Formulations of stability conditions given above rely on a global quadratic Lyapunov function. In many cases, weaker conditions can be obtained when a piecewise-quadratic Lyapunov function is applied instead of the global one, as it was shown in Theorem 2.2. To apply this result to the state-feedback case, the space of the process states partitioned into constant activation cells $S_l$, $l = 1, ..., L$, must be used, as for the uncontrolled process before. Recalling that $K(l)$ denotes a set containing indexes of all fuzzy rules with non-zero activation level within $S_l$, the state-feedback controller takes the form (compare with (2.22))

$$u(k) = -\sum_{j \in K(l)} \tilde{w}^j(k)\mathbf{F}_j x(k), \quad x(k) \in S_l, \quad l \in L \tag{2.32}$$

The closed-loop system dynamics can then be described as follows

$$x(k+1) = \sum_{i \in K(l)} \tilde{w}^i(k)[\mathbf{A}_i - \mathbf{B}_i \sum_{j \in K(l)} \tilde{w}^j(k)\mathbf{F}_j]x(k), \quad x(k) \in S_l, \ l \in L \tag{2.33}$$

where $\sum_{j \in K(l)} \tilde{w}^j(k) = 1$. Due to this last equality, (2.33) can be written in the form

$$x(k+1) = \sum_{i \in K(l)} \sum_{j \in K(l)} \tilde{w}^i(k)\tilde{w}^j(k)[\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_j]x(k), \quad x(k) \in S_l, \ l \in L \tag{2.34}$$

which is a partitioned form of the overall dynamics description (2.25), corresponding to the partitioning of the whole process domain into L constant activation cells $S_l$. Theorem 2.2 can now be directly applied to the considered state-feedback case yielding the following result.

**Corollary 2.5** *Equilibrium point of the control system (2.34) described by the rules of the process model (2.8) and of the controller (2.21) is globally asymptotically stable, if there exists L symmetric and positive definite matrices* $\mathbf{P}_l$, $l \in L$ *such that the following set of inequalities is satisfied*

$$[\mathbf{A}_i^T - \mathbf{B}_i\mathbf{F}_j]\mathbf{P}_m[\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_j] - \mathbf{P}_l < 0, \quad \text{for every } (l,m) \in \Omega, \quad i,j \in K(l) \tag{2.35}$$

*where $\Omega$ denotes the set of index pairs indicating all possible one-step closed-loop system state transitions between the activation cells $S_l$, see (2.15).* □

The weak point of the above formulation is the necessity to define, in advance, all possible one-step state transitions between all activation cells, *i.e.*, to define the set $\Omega$. It should be emphasized that possible one-step state transitions for the closed-loop control system may be different than those for the uncontrolled process, and obviously the former must be taken into account in the definition of the set $\Omega$ in the above corollary. Certainly, underestimation of this set may be dangerous as omitted transitions may be those causing instability. Therefore, overestimation is very likely, leading to an increased, large number of inequalities in (2.35) – and the more inequalities the more constraining is the stability condition.

The design and analysis presented so far was for closed-loop TS fuzzy control systems with locally designed state-feedback matrices, as presented in the initial part of Section 2.2. This approach seems to be a convincing and natural generalization of the classical state-feedback design for a linear process model at a given equilibrium point. But another, more global oriented approach is also possible. It is based on a design of state-feedback matrices $\mathbf{F}_i$ ensuring stability of the overall state-feedback control system on the basis of a solution of a system of linear matrix inequalities, with respect to both state-feedback and stability related (Lyapunov type) matrices. For instance, with respect to all matrices $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{F}_i$, $i = 1, ..., r$ given in Theorem 2.4 – the system of nonlinear inequalities (2.30)-(2.31) is then reformulated to the form of a system of linear matrix inequalities [132]. In [151] a similar approach is proposed for the design of a state-feedback $H_\infty$ fuzzy controller, based on a piecewise-quadratic Lyapunov function. The sets of linear matrix inequalities can then be effectively solved using available packages, *e.g.*, *LMI Toolbox* of the MATLAB® package could be applied. We shall not present this approach here in detail, as in the author's opinion a more natural, more intuitive approach is the decentralized design methodology of a TS fuzzy controller, based on a number of local feedback designs. Moreover, in the case of an unsatisfactory result of the design, not only values of the feedback coefficients should be corrected, but first of all the number, positioning and shape of membership functions.

In practical applications, an entire vector of state variables may not be available for measurement. In this case the theory of state-feedback control suggests the use of a state observer. This approach can also be applied when designing a control system with a state-feedback TS fuzzy controller, constructing a state observer as a TS fuzzy system with the structure of rules identical to that of the rules of the process model and of the controller – but with rule consequents being classical formulae of linear state observers. Because controllers with feedback from observed state have not yet found po-

pularity in process industries, we shall omit the presentation of the design of fuzzy state observers and stability analysis of the closed-loop control systems with observers, asking the reader to refer to the literature [132].

**Example 2.5**
Let us reconsider a TS fuzzy model of a process presented in Example 2.4, but with a control variable $u$ added. Fuzzy sets $X_1$ and $X_2$ defined by trapezoidal membership functions are presented in Fig. 2.19, while rules of the process model can be presented in the following form

$$\begin{aligned} \mathrm{R}_p^1: &\quad \text{IF} \quad x_2(k) \text{ is } X_1 \quad \text{THEN} \quad x^1(k+1) = \mathbf{A}_1 x(k) + \mathbf{B}u(k) \\ \mathrm{R}_p^2: &\quad \text{IF} \quad x_2(k) \text{ is } X_2 \quad \text{THEN} \quad x^2(k+1) = \mathbf{A}_2 x(k) + \mathbf{B}u(k) \end{aligned}$$

where $x(k) = [x_1(k)\ x_2(k)]^T \in \mathbb{R}^2$, $u(k) \in \mathbb{R}$,

$$\mathbf{A}_1 = \begin{bmatrix} 1 & -0.5 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} -1 & -0.5 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

It was shown in Example 2.4 that the autonomous (uncontrolled) TS fuzzy system can generate, for certain initial conditions, unstable state trajectories. We shall now design, for that system, a TS fuzzy controller with state-feedback ensuring stable operation of the closed-loop control system. The controller rules will have the following form

$$\begin{aligned} \mathrm{R}_c^1: &\quad \text{IF} \quad x_2(k) \text{ is } X_1 \quad \text{THEN} \quad u(k) = -\mathbf{F}_1 x(k) \\ \mathrm{R}_c^2: &\quad \text{IF} \quad x_2(k) \text{ is } X_2 \quad \text{THEN} \quad u(k) = -\mathbf{F}_2 x(k) \end{aligned}$$

It will be most simple to assume identical placement of eigenvalues of state matrices $\mathbf{A}_1 - \mathbf{B}\mathbf{F}_1$ and $\mathbf{A}_2 - \mathbf{B}\mathbf{F}_2$ of individual local feedback systems. Assuming the eigenvalues equal to 0.5 and 0.4 the elements of vectors $\mathbf{F}_1$ and $\mathbf{F}_2$ were calculated (using the *place* function from the *Control System Toolbox* of the MATLAB® package).

$$\mathbf{F}_1 = \begin{bmatrix} 0.1 & -0.3 \end{bmatrix}, \quad \mathbf{F}_2 = \begin{bmatrix} -1.9 & -0.3 \end{bmatrix}$$

The situation of identical rules in the fuzzy controller and in the process model used for the simulation of the control system will be considered. Thus, to examine the stability it is enough to calculate three matrices, which due to the assumptions should have identical values

$$\begin{aligned} \mathbf{A}_1 - \mathbf{B}\mathbf{F}_1 &= \mathbf{A}_2 - \mathbf{B}\mathbf{F}_2 = \\ &= \mathbf{D}_{12} = \frac{1}{2}[(\mathbf{A}_1 - \mathbf{B}\mathbf{F}_2) + (\mathbf{A}_2 - \mathbf{B}\mathbf{F}_1)] = \begin{bmatrix} 0.9 & -0.2 \\ 1 & 0 \end{bmatrix} \end{aligned}$$

The closed-loop TS fuzzy control system will be, therefore, described by one equation

$$x(k+1) = \mathbf{D}_{12}x(k)$$

and is, of course, stable with assumed eigenvalues of the state matrices equal to 0.5 and 0.4.

For a more elaborate illustration of the stability properties of the closed-loop system we shall repeat the design of the controller assuming a slightly different vector of feedback coefficients in the second subprocess, namely $\mathbf{F}_2 = [-2.1 \ -0.2]$, which corresponds to eigenvalues of the matrix $\mathbf{A}_2 - \mathbf{BF}_2$ equal to 0.5 and 0.6. Now we have

$$\mathbf{A}_1 - \mathbf{BF}_1 = \begin{bmatrix} 0.9 & -0.2 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{A}_2 - \mathbf{BF}_2 = \begin{bmatrix} 1.1 & -0.3 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{D}_{12} = \frac{1}{2}[(\mathbf{A}_1 - \mathbf{BF}_2) + (\mathbf{A}_2 - \mathbf{BF}_1)] = \begin{bmatrix} 1 & -0.25 \\ 1 & 0 \end{bmatrix}$$

A sufficient condition of stability of the closed-loop system is, according to the Corollary 2.3, the existence of a symmetric, positive definite matrix $\mathbf{P}$ satisfying the system of inequalities

$$(\mathbf{A}_1 - \mathbf{BF}_1)^T\mathbf{P}(\mathbf{A}_1 - \mathbf{BF}_1) - \mathbf{P} < \mathbf{0}$$
$$(\mathbf{A}_2 - \mathbf{BF}_2)^T\mathbf{P}(\mathbf{A}_2 - \mathbf{BF}_2) - \mathbf{P} < \mathbf{0}$$
$$(\mathbf{D}_{12})^T\mathbf{P}\mathbf{D}_{12} - \mathbf{P} < \mathbf{0}$$

Using the *LMI Toolbox* of the MATLAB® package it was checked that such a matrix exists, obtaining

$$\mathbf{P} = \begin{bmatrix} 1.9755 & -0.7968 \\ -0.7968 & 0.9459 \end{bmatrix}$$

□

### 2.2.2 Discrete TS Fuzzy Output-feedback Controllers

In control systems of technological processes linear controllers with feedback from the output are usually used, particularly the well known PID type controllers. Along with a more and more wide application of multilayer control structures with on-line optimization of operating points, there grows a necessity of applications of nonlinear controllers, which are able to operate properly for a wide range of different set-points. Thus, in industrial applications it is important to have a relatively simple nonlinear controller, which would ideally be a nonlinear generalization of the PID controller. The construction of a discrete TS fuzzy output-feedback controller fulfilling the above postulates was proposed in [31], see also [33, 32], where it was called a *multi-regional controller*.

Let us consider a TS fuzzy model of a process which will be used for the design of a fuzzy controller. We shall assume the rules of this model in the following form

$$\begin{aligned} \mathrm{R}_p^i : \mathrm{IF} \quad & y(k) \text{ is } A_0^i \text{ and } y(k-1) \text{ is } A_1^i \text{ and } \cdots \text{ and } y(k-n_R) \text{ is } A_{n_R}^i \\ & \text{and } u(k-1) \text{ is } B_1^i \text{ and } \cdots \text{ and } u(k-m_R) \text{ is } B_{m_R}^i \\ \text{THEN} \quad & y^i(k+1) = a_1^i y(k) + a_2^i y(k-1) + \cdots + a_{n_A}^i y(k-n_A+1) + \\ & + b_0^i u(k) + b_1^i u(k-1) + \cdots + b_{m_B}^i u(k-m_B) \end{aligned} \tag{2.36}$$

where $i = 1, \ldots, r$ indexes rules and related fuzzy sets (sub-domains) in the domain of the TS fuzzy model (generally multi-dimensional), $y(k)$ is a value of the process output at the $k$-th sampling instant, $u(k)$ is a value of the process control input at the $k$-th sampling instant, $A_j^i \in \mathbb{Y}_j$, $B_j^i \in \mathbb{U}_j$, whereas $a_j^i$ and $b_j^i$ are coefficients of functions in the rule consequents, $i = 1, ..., r$. Elements of each of the sets $\mathbb{Y}_j = \{Y_{j1}, ..., Y_{jr_{yj}}\}$ are fuzzy sets covering the domain of the variable $y(k-j)$, $j = 0, \ldots, n_R$, similarly $\mathbb{U}_j = \{U_{j1}, ..., U_{jr_{uj}}\}$ for $u(k-j)$, $j = 1, \ldots, m_R$.

The presented situation is the most general one; we usually have to deal with a far simpler case, when

$$\mathbb{Y}_0 = \cdots = \mathbb{Y}_{n_R} = \mathbb{Y}, \quad \mathbb{Y} = \{Y_1, ..., Y_{r_y}\}$$

*i.e.*, partitions of the domain of each of the current and past output variables, *i.e.*, $y(k)$, $y(k-1), ..., y(k-n_R)$, are the same. Similarly, the partitions of the domain of the control input vector are usually the same, $\mathbb{U}_1 = \cdots = \mathbb{U}_{m_R} = \mathbb{U}$, $\mathbb{U} = \{U_1, ..., U_{r_u}\}$.

Dynamic linear models applied in the consequents of rules (2.36) are the ARX-type models. These models also apply to systems with time delays equal to several, say $\tau$, sampling periods. Then, appropriate coefficients of the rule consequents will only be zero, $b_0^i = b_1^i = \cdots = b_\tau^i = 0$.

The output of a fuzzy model will be calculated according to the general formula (2.5), *i.e.*,

$$y(k+1) = \frac{\sum_{i=1}^{r} w^i(k)\, y^i(k+1)}{\sum_{l=1}^{r} w^l(k)} \tag{2.37}$$

where $w^i(k)$ are activation levels of individual rules (2.36) at sampling instant $k$,

$$w^i(k) = \prod_{j=0}^{n_R} \mu_{A_j^i}(y(k-j)) \prod_{p=1}^{m_R} \mu_{B_p^i}(u(k-p)) \tag{2.38}$$

For a model of the process described by the rules (2.36) the following *rules of a TS fuzzy controller* are formulated

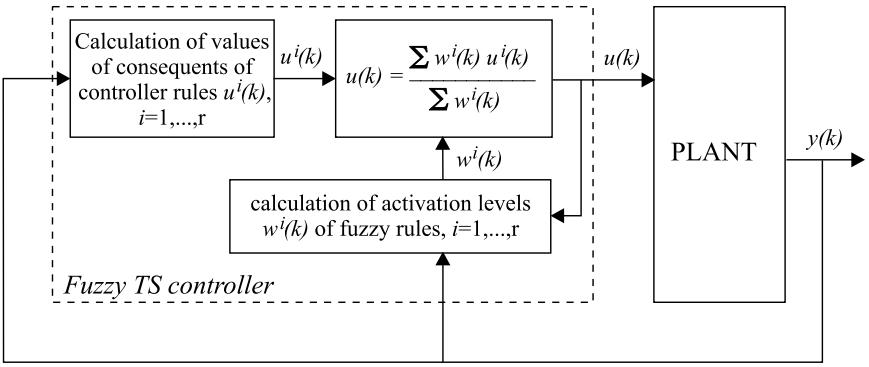


**Fig. 2.22.** Structure of a control system with the TS fuzzy output-feedback controller

$$\begin{aligned} \mathrm{R}_c^j:\ \text{IF}\ \ & y(k) \text{ is } A_0^j \ \text{ and } \ y(k-1) \text{ is } A_1^j \ \text{ and } \cdots \text{ and } \ y(k-n_R) \text{ is } A_{n_R}^j \\ & \text{and } \ u(k-1) \text{ is } B_1^j \ \text{ and } \cdots \text{ and } \ u(k-m_R) \text{ is } B_{m_R}^j \\ \text{THEN } \ & u^j(k) = c_1^j e(k) + c_2^j e(k-1) + \cdots + c_{n_C}^j e(k-n_C+1) + \\ & + d_1^j u(k-1) + d_2^j u(k-2) + \cdots + d_{m_D}^j u(k-m_D) \end{aligned} \quad (2.39)$$

where $j = 1, \ldots, r$ indexes the rules whose antecedents are identical to those in the model (2.36), $e(k) = y^{sp}(k) - y(k)$ is the control error at sampling instant $k$, while $c_j^j$ and $d_j^j$ are coefficients of functions in the rule consequents. The form of the discrete control algorithm occurring in the rule consequents (2.39) is fairly general, its special cases are algorithms of *discrete PID controllers* or unconstrained (explicit) versions of *predictive controllers of DMC and GPC type* (described in Chapter 3).

The controller output equation takes a standard form for TS fuzzy structures:

$$u(k) = \frac{\sum_{j=1}^{r} w^j(k)\, u^j(k)}{\sum_{l=1}^{r} w^l(k)} = \sum_{j=1}^{r} \tilde{w}^j(k)\, u^j(k) \quad (2.40)$$

where $w^j(k)$ are activation levels of individual rules (2.39) at $k$-th sampling instant, defined by formulae (2.38), due to the same rule antecedents as in (2.36). The structure of a control system with the TS fuzzy output-feedback controller described above is presented in Fig. 2.22.

Let us note that in the rule antecedents of the controller (2.39) the condition "$u(k)$ is $B_0^j$" does not appear, as this would lead to a logical inconsistency. Because a control value $u(k)$ for $k$-th sampling instant is to be calculated in functional consequents of the rules of the controller, then $u(k)$ cannot occur at $k$-th sampling instant in the rule antecedents. Thus, the simplest solution is to design the antecedents of the controller rules without the condition

"$u(k)$ is $B_0^j$", as it was done in (2.39). This is natural, especially for the widely applied discrete control with the extrapolator of a zero order. Since we measure and thus know the value of the process output $y(k)$ just before calculating the control value $u(k)$ at the $k$-th sampling instant (*i.e.*, for the sampling period beginning at that moment), but the process is still under the influence of the control $u(k-1)$ when $u(k)$ is being computed.

For simulative or analytical research of the closed-loop control system with the fuzzy controller described by (2.39), presented in Fig. 2.22, it is possible to use a process model different that given by the rules (2.36) and used for the construction of the controller. In particular, it can be a slightly different TS fuzzy model with the number of rules $ro$ (generally $ro \neq r$) and the rules in the form

$$\begin{aligned} \mathrm{R}_{po}^i: \ &\text{IF} \ \ y(k) \text{ is } A_0^i \ \text{ and } \ y(k-1) \text{ is } A_1^i \ \text{ and } \cdots \text{ and } \ y(k-n_O) \text{ is } A_{n_O}^i \\ &\text{and} \ \ u(k) \text{ is } B_0^i \ \text{ and } \ u(k-1) \text{ is } B_1^i \ \text{ and } \cdots \text{ and } \ u(k-m_O) \text{ is } B_{m_O}^i \\ &\text{THEN} \ \ y^i(k+1) = a_1^i y(k) + a_2^i y(k-1) + \cdots + a_{n_A}^i y(k-n_A+1) + \\ &\qquad + b_0^i u(k) + b_1^i u(k-1) + \cdots + b_{m_B}^i u(k-m_B) \end{aligned} \tag{2.41}$$

where $i = 1, ..., ro$, while the parameters $n_A$ and $m_B$ of the rules (2.36) and (2.41) can be of different values. We did not introduce different descriptions for those parameters, in order not to complicate the notation; it does not lead to misunderstandings as in the closed-loop control system the description of the process model (2.41) only is present. The activation levels of the rules (2.41) will be denoted by $w_o^i(k)$, and their normalized values by $\tilde{w}_o^i(k)$,

$$y(k+1) = \frac{\sum_{i=1}^{ro} w_o^i(k)\, y^i(k+1)}{\sum_{l=1}^{ro} w_o^l(k)} = \sum_{i=1}^{ro} \tilde{w}_o^i(k)\, y^i(k+1) \tag{2.42}$$

Let us note that in the rule antecedents of the process model (2.41) the condition "$u(k)$ is $B_0^i$" can be present, in general. It does not lead to logical inconsistency here, as in the consequents of these rules the process output value for the next ($(k+1)$-th) sampling instant is calculated.

However, if it is desired that in the rule antecedents of the controller the condition "$u(k)$ is $B_0^j$" should be present, then it is possible provided modified consequents in the controller rules are designed. Namely, in order to eliminate the logical inconsistency, it is necessary to apply in the rule consequents control algorithms calculating the control for the next sampling period (see [161, 32]), *i.e.*, to use the controller rules of the following form

$$\begin{aligned} \mathrm{R}_c^j: \ &\text{IF} \ \ y(k) \text{ is } A_0^j \ \text{ and } \ y(k-1) \text{ is } A_1^j \ \text{ and } \cdots \text{ and } \ y(k-n_R) \text{ is } A_{n_R}^j \\ &\text{and} \ \ u(k) \text{ is } B_0^j \ \text{ and } \ u(k-1) \text{ is } B_1^j \ \text{ and } \cdots \text{ and } \ u(k-m_R) \text{ is } B_{m_R}^j \\ &\text{THEN} \ \ u^j(k+1) = c_1^j e(k) + c_2^j e(k-1) + \cdots + c_{n_C}^j e(k-n_C+1) + \\ &\qquad + d_1^j u(k) + d_2^j u(k-1) + d_3^j u(k-2) + \cdots + d_{m_D}^j u(k-m_D+1) \end{aligned} \tag{2.43}$$

It should be added that conditions containing the process control input variables are often not present at all in the antecedents of the process model rules. Therefore, consideration of the problem of occurrence of the condition "$u(k)$ is $B_0^i$" in the rule antecedents is of no significance for these applications. The key variables of the rule antecedents of a TS fuzzy model are usually the process output variables $y(k), y(k-1), \ldots$ . Therefore, for an important class of problems it is enough to use a process model (and thus a controller) in which conditions of the antecedents are defined for the output variables only, *i.e.*, to use the following rules

$$\begin{aligned} \mathrm{R}_p^i: \ \text{IF} \ \ & y(k) \text{ is } A_0^i \ \text{ and } \ y(k-1) \text{ is } A_1^i \ \text{ and } \cdots \text{ and } \ y(k-n_R) \text{ is } A_{n_R}^i \\ \text{THEN} \ \ & y^i(k+1) = a_1^i y(k) + a_2^i y(k-1) + \cdots + a_{n_A}^i y(k-n_A+1) + \\ & + b_0^i u(k) + b_1^i u(k-1) + \cdots + b_{m_B}^i u(k-m_B) \end{aligned} \tag{2.44}$$

Let us now consider the stability of a closed-loop control system with a TS fuzzy output-feedback controller and a process model in the form of a TS fuzzy system. Let us begin from the general case when the model used for describing the process in the feedback loop, although also a TS fuzzy one, is however different than the one used for the design of the controller. This generally results in a different number and different antecedents of process and controller fuzzy rules. Let us assume the process model with the rules (2.41) and the controller with the rules (2.39). To shorten the notation, in all the formulae presented next we shall denote levels of activation of the process and controller rules by $w_o^i$, $w^j$ instead of $w_o^i(k)$, $w^j(k)$ (dropping the time variable $k$), and consequently, the normalized levels of activation, see (2.6), by $\tilde{w}_o^i$, $\tilde{w}^j$. For the simplicity and clarity of the following algebraic expressions we shall also assume that

$$\begin{aligned} n_A &= n_C = n \\ m_B &= m_D = m \end{aligned} \tag{2.45}$$

which should be understood as follows:

- $n = \max\{n_A, n_C\}$ and the missing coefficients $a_p^i$ (if $n_A < n_C$) or $c_p^i$ (if $n_A > n_C$) are zeros; similarly
- $m = \max\{m_B, m_D\}$ and the missing coefficients $b_p^i$ (if $m_B < m_D$) or $d_p^i$ (if $m_B > m_D$) are zeros.

Substituting dependencies describing local process models into (2.42) we have

$$y(k+1) = \sum_{i=1}^{ro} \tilde{w}_o^i \left\{ \sum_{p=1}^{n_A} a_p^i y(k-p+1) + \sum_{q=0}^{m_B} b_q^i u(k-q) \right\} \tag{2.46}$$

Analogously, inserting equations of local controllers (2.39) into (2.40) we obtain

$$u(k) = \sum_{j=1}^{r} \tilde{w}^j \left\{ \sum_{t=1}^{n_C} c_t^j e(k-t+1) + \sum_{s=1}^{m_D} d_s^j u(k-s) \right\}$$
$$= \sum_{j=1}^{r} \tilde{w}^j \left\{ \sum_{t=1}^{n_C} c_t^j y^{sp}(k-t+1) - \sum_{t=1}^{n_C} c_t^j y(k-t+1) + \sum_{s=1}^{m_D} d_s^j u(k-s) \right\} \tag{2.47}$$

Now inserting this dependence into (2.46) and taking into account that $\sum_{j=1}^{r} \tilde{w}^j = 1$, we obtain the equation describing the closed-loop control system

$$y(k+1) = \sum_{i=1}^{ro} \tilde{w}_o^i \sum_{j=1}^{r} \tilde{w}^j \left\{ \sum_{p=1}^{n_A} a_p^i y(k-p+1) + \sum_{q=1}^{m_B} b_q^i u(k-q) \right\} +$$
$$+ \sum_{i=1}^{ro} \tilde{w}_o^i b_0^i \sum_{j=1}^{r} \tilde{w}^j \left\{ \sum_{t=1}^{n_C} c_t^j y^{sp}(k-t+1) - \sum_{t=1}^{n_C} c_t^j y(k-t+1) + \right.$$
$$\left. + \sum_{s=1}^{m_D} d_s^j u(k-s) \right\}$$

Applying now the assumption (2.45) which allows for a consistent and clear notation we can, after proper grouping of the terms, present the obtained equation in the following form

$$y(k+1) = \sum_{i=1}^{r} \tilde{w}_o^i \sum_{j=1}^{r} \tilde{w}^j \left\{ \sum_{p=1}^{n} (a_p^i - b_0^i c_p^j) y(k-p+1) + \right.$$
$$\left. + \sum_{q=1}^{m} (b_q^i + b_0^i d_q^j) u(k-q) + \sum_{t=1}^{n} b_0^i c_t^j y^{sp}(k-t+1) \right\} \tag{2.48}$$

Let us now define a state vector $x(k) \in \mathbb{R}^{n+m-1}$ describing the dynamics given by (2.48) and (2.47) as follows

$$x(k) = [y(k)\ y(k-1)\ \cdots\ y(k-n+1)\ u(k-1)\ u(k-2)\ \cdots\ u(k-m)]^T \tag{2.49}$$

Furthermore, in order to have the following expressions concise, let us define the parameters

$$ac_p^{i,j} = a_p^i - b_0^i c_p^j, \quad p = 1, ..., n$$
$$bd_q^{i,j} = b_q^i + b_0^i d_q^j, \quad q = 1, ..., m$$

Then, taking into account that (2.47) is equivalent to the equation

$$u(k) = \sum_{i=1}^{ro} \tilde{w}_o^i \sum_{j=1}^{r} \tilde{w}^j \left\{ \sum_{t=1}^{n} c_t^j [y^{sp}(k-t+1) - y(k-t+1)] + \right.$$
$$\left. + \sum_{s=1}^{m} d_s^j u(k-s) \right\}$$

the description of the closed-loop control system can be presented in the form

$$x\left(k+1\right)=\sum_{i=1}^{ro}\sum_{j=1}^{r}\tilde{w}_o^i\tilde{w}^j\mathbf{A}_{ij}x(k)+\sum_{i=1}^{ro}\sum_{j=1}^{r}\tilde{w}_o^i\tilde{w}^j\sum_{t=1}^{n}\mathbf{E}_{ij}\tilde{y}^{sp}(k) \tag{2.50}$$

where

$$\mathbf{A}_{ij}=\begin{bmatrix} ac_1^{i,j} & ac_2^{i,j} & \cdots & ac_{n-1}^{i,j} & ac_n^{i,j} & bd_1^{i,j} & bd_2^{i,j} & \cdots & bd_{m-1}^{i,j} & bd_m^{i,j} \\ 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ -c_1^j & -c_2^j & \cdots & -c_{n-1}^j & -c_n^j & d_1^j & d_2^j & \cdots & d_{m-1}^j & d_m^j \\ 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{2.51}$$

$$\mathbf{E}_{ij}=\begin{bmatrix} b_0^ic_1^j & b_0^ic_2^j & \cdots & b_0^ic_{n-1}^j & b_0^ic_n^j \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ c_1^j & c_2^j & \cdots & c_{n-1}^j & c_n^j \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{2.52}$$

and $\tilde{y}^{sp}(k)$ is defined as

$$\tilde{y}^{sp}(k)=[y^{sp}(k)\;\; y^{sp}(k-1)\;\; \cdots\;\; y^{sp}(k-n+1)]^T \tag{2.53}$$

Dynamics of the autonomous system is described by the equation

$$x\left(k+1\right)=\sum_{i=1}^{ro}\sum_{j=1}^{r}\tilde{w}_o^i(k)\tilde{w}^j(k)\mathbf{A}_{ij}x(k) \tag{2.54}$$

Applying now to (2.54) the Lyapunov theorem we can obtain a result analogous to Theorem 2.1, which can also be treated as a direct application of this theorem, provided all matrices $\mathbf{A}_{ij}$ with indexes for which $w_o^i(k)w^j(k)=0$ are always excluded:

**Corollary 2.6** *The equilibrium point of a TS fuzzy system described by (2.54) is globally asymptotically stable, if there exists a symmetric positive definite matrix* $\mathbf{P}$ *such that for each matrix* $\mathbf{A}_{ij}$ *the following equation is fulfilled*

$$\mathbf{A}_{ij}^T \mathbf{P} \mathbf{A}_{ij} - \mathbf{P} < \mathbf{0} \tag{2.55}$$

*for all pairs* $(i,j)$, $i = 1,2,...,ro$, $j = 1,2,...,r$ *except for those for which always* $w_o^i(k)w^j(k) = 0$. □

If we assume that the number of rules and the rule antecedents in a model used for the description of the process in a closed-loop control system and in the controller description are identical, then the following equalities are true: $ro = r$, $\tilde{w}_o^i(k) = \tilde{w}^i(k)$, $i = 1,...,r$ and $\tilde{w}^i(k)\tilde{w}^j(k) = \tilde{w}^j(k)\tilde{w}^i(k)$, for every value of $i,j = 1,2,...,r$. Using this last equality it is possible to diminish the number of matrices $\mathbf{A}_{ij}$ occurring in (2.55), proceeding analogously as in the previous section. Namely, the dynamics (2.54) can be then presented in the form

$$x\,(k+1) = \left[ \sum_{i=1}^{r} \tilde{w}^i(k)\tilde{w}^i(k)\mathbf{A}_{ii} + 2 \sum_{i,j=1,\, i<j}^{r} \tilde{w}^i(k)\tilde{w}^j(k)\overline{\mathbf{A}_{ij}} \right] x(k) \tag{2.56}$$

where

$$\overline{\mathbf{A}_{ij}} = \frac{\mathbf{A}_{ij} + \mathbf{A}_{ji}}{2}, \quad i < j,\ i,j = 1,2,...,r$$

while $\sum_{i,j=1,\, i<j}^{r}$ denotes addition of all terms with pairs of indexes $i,j$ such that $i < j$. In stability conditions (2.55) it is now enough to consider, instead of matrices $\mathbf{A}_{ij}$, $i,j = 1,2,...,r$, only the matrices $\mathbf{A}_{ii}$, $i = 1,...,r$ and $\overline{\mathbf{A}_{ij}}$, $i < j$, $i,j = 1,2,...,r$. Conditions (2.55) can in this case be made slightly weaker, by application of Theorem 2.4 to the dynamic system (2.56).

**Corollary 2.7** *Let the maximum number of rules activated at the same sampling instant be no higher than* $s$, $1 < s \leq r$. *Then the equilibrium point of the dynamic system described by (2.56) is globally asymptotically stable, if there exist a positive definite matrix* $\mathbf{P}$ *and a positive semi-definite matrix* $\mathbf{Q}$, *satisfying the following conditions*

$$\begin{aligned} {\mathbf{A}_{ii}}^T \mathbf{P} \mathbf{A}_{ii} - \mathbf{P} + (s-1)\mathbf{Q} < 0, \quad i = 1,2,...,r \\ \overline{\mathbf{A}_{ij}}^T \mathbf{P} \overline{\mathbf{A}_{ij}} - \mathbf{P} - \mathbf{Q} \leq 0, \quad i < j,\ i,j = 1,2,...,r \end{aligned}$$

*for all pairs* $(i,j)$ *except those for which* $w^i(k)w^j(k) = 0$ *for every value of* $k$. □

The performed stability analysis does not directly include the case of a control system with a controller defined by rules (2.43), whose antecedents additionally contain the condition "$u(k)$ is $B_0^j$", and thus consequents are defined by slightly different formulae – compare (2.39) with (2.43). A stability

analysis of such a control system can be conducted in a way identical as was done above, obtaining analogous results – only matrices $\mathbf{A}_{ij}$ and $\mathbf{E}_{ij}$ will be defined by slightly different formulae. These formulae can be obtained directly from those already mentioned, only a modification of the coefficients of the controller rule consequents is necessary, from the form

$$u^j(k+1) = c_1^j e(k) + c_2^j e(k-1) + \cdots + c_{n_C}^j e(k-n_C+1) + \\ + d_1^j u(k) + d_2^j u(k-1) + \cdots + d_{m_D}^j u(k-m_D+1)$$

to the form

$$u^j(k) = \bar{c}_1^j e(k) + \bar{c}_2^j e(k-1) + \cdots + \bar{c}_{\bar{n}_C}^j e(k-\bar{n}_C+1) + \\ + d_1^j u(k-1) + d_2^j u(k-3) + \cdots + d_{m_D}^j u(k-m_D)$$

where $\bar{n}_C = n_C + 1$, $\bar{c}_1^j = 0$, $\bar{c}_p^j = c_{p-1}^i$, $p = 2, ..., n_C + 1$. Moreover, $w^j(k-1)$ should be, consequently, inserted to all formulae in place of $w^j(k)$, $j = 1, ..., r$.

**Example 2.6**
Let us consider a process model described by the following fuzzy rules in the form (2.44)

$$\mathrm{R}_p^1: \quad \text{IF} \quad y(k) \text{ is } Y_1 \quad \text{THEN} \quad y^1(k+1) = 0.7y(k) + 0.8u(k) \tag{2.57}$$

$$\mathrm{R}_p^2: \quad \text{IF} \quad y(k) \text{ is } Y_2 \quad \text{THEN} \quad y^2(k+1) = 0.3y(k) + 0.2u(k) \tag{2.58}$$

Fig. 2.23 (a) presents fuzzy sets $Y_1$ and $Y_2$, while Fig. 2.23 (b) presents step responses of local processes as described by consequents of the rules (2.57) and (2.58). Comparison of these responses shows that the considered process, despite its simple structure, is strongly nonlinear – nonlinear is in its statics

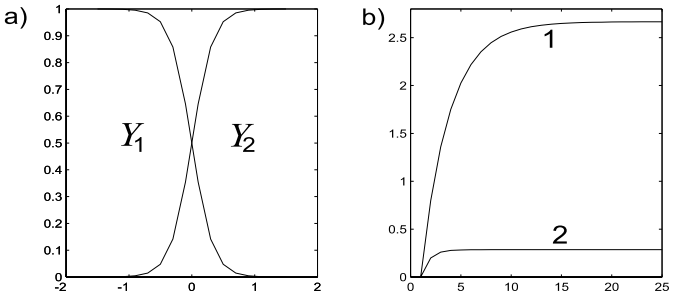


**Fig. 2.23.** a) sigmoidal membership functions, b) step responses, of local processes in Example 2.6

(gain) as well as in dynamics (time constant). On the other hand, it is not easily controlled, due to its time constant being relatively small when compared to the sampling period.

Discrete PI controllers were designed for process models from the rule consequents (2.57) and (2.58), in a version with integration implemented by the method of trapezoids, *i.e.*, in the following form

$$u(k) = u(k-1) + k_p(1 + \frac{T_p}{2T_I})e(k) + k_p(-1 + \frac{T_p}{2T_I})e(k-1)$$

where $k_p$ is the controller gain and $T_I$ is its integral time. Values of these parameters were selected by Ziegler-Nichols rules (see *e.g.*, [3]), followed by slight corrections, resulting in

$$k_p^1 = 0.9, \;\; T_I^1 = 3$$
$$k_p^2 = 2.0, \;\; T_I^2 = 1$$

Individual PI controllers operate very well in vicinities of their operating points, but poorly within the whole domain of the output variable. Figure 2.24 presents trajectories in the control system with a nonlinear process and, subsequently, with the first PI controller designed for the operating point $y = -1$ and with the second PI controller designed for the operating point $y = +1$. The first controller is definitely too slow in the range of positive values of $y$.

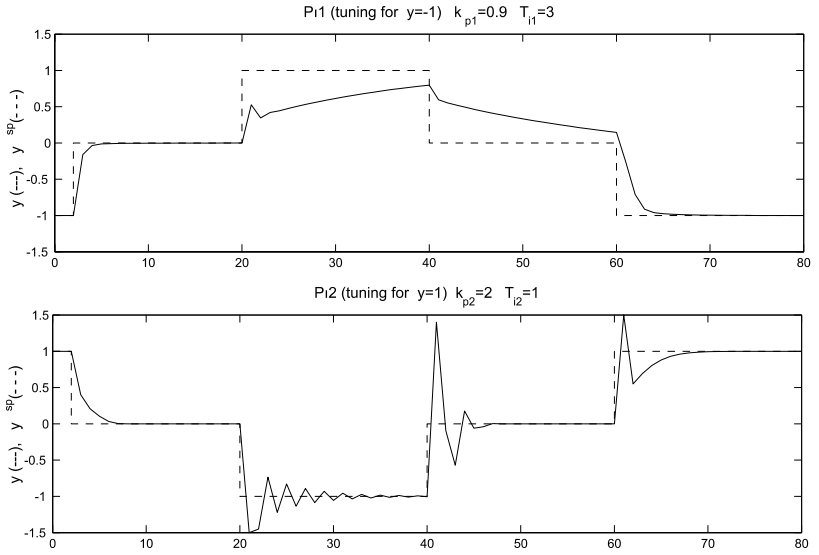


**Fig. 2.24.** Controlled variable trajectories in control systems with a nonlinear fuzzy process and the PI controller designed for the operating point $y = -1$ (upper figure) and for the operating point $y = +1$ (lower figure)

The second, in turn, is too aggressive, its trajectories are too oscillating not only for negative values of the controlled variable $y$, but also in the vicinity of its zero value. Moreover, Fig. 2.24 presents a trajectory which is too aggressive even in the vicinity of its own operating point $y = +1$, if a step change of the set-point from zero is followed. Therefore, before going on to the construction of a TS fuzzy controller the gain of the second controller was decreased to $k_p^2 = 1.5$.

Assuming antecedents such as in the process model and consequents in the form of the discussed local PI controllers we obtain the following rules of a TS fuzzy controller

$$\mathrm{R}_c^1\text{: IF } y(k) \text{ is } Y_1 \text{ THEN } u^1(k) = u(k-1) + 1.05e(k) - 0.75e(k-1) \tag{2.59}$$

$$\mathrm{R}_c^2\text{: IF } y(k) \text{ is } Y_2 \text{ THEN } u^2(k) = u(k-1) + 2.25e(k) - 0.75e(k-1) \tag{2.60}$$

Figure 2.25 presents trajectories of the process output and the control input in the closed-loop system with the designed TS fuzzy controller, for a fairly wide range of changes of the set-point, while Fig. 2.26 presents trajectories of the output variable in a slightly narrower, but critical range of changes of the set-point between $-1$ and 1. In the lower part of Fig. 2.26 a trajectory for the case $k_p^2 = 2$ is presented, namely without the previously mentioned damping of the gain $k_p^2$. As could be expected, increasing gain coefficient $k_p^2$ slightly

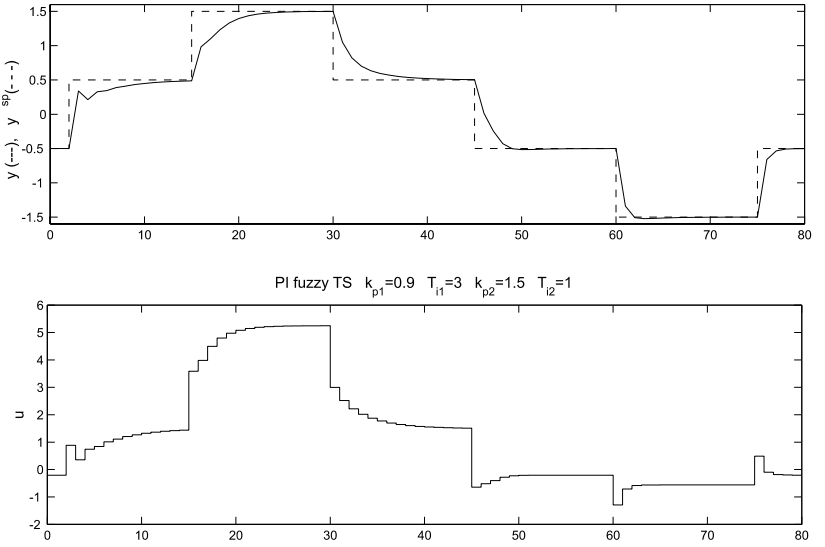


**Fig. 2.25.** Trajectories of the process output and input variables in the control system with the TS fuzzy controller

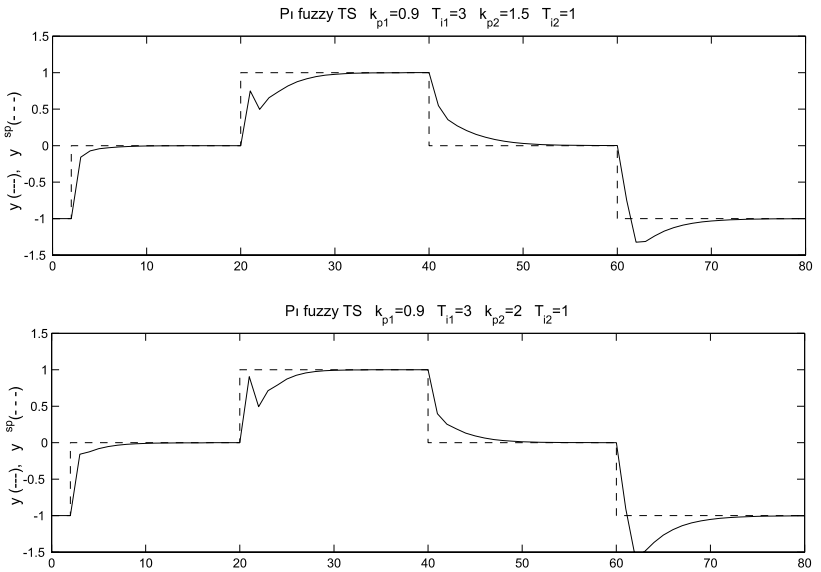


**Fig. 2.26.** Trajectories of the output variable in the control system with the PI fuzzy controller: with $k_p^2 = 1.5$ (upper trajectory), $k_p^2 = 2$ (lower trajectory)

speeds up the operation of a control system in the range of positive output values. Unfortunately, at the same time the overshoot increases with changes of the set-point in the direction of negative values, which can clearly be seen for a set-point step from zero to $-1$. Therefore, it justifies the damping of $k_p^2$ carried out previously. In Chapter 3 we shall present that the described conflict can be significantly reduced by employing a nonlinear predictive controller, with faster operation for positive output values and a simultaneous reduction of the overshoot for the negative values.

To perform stability analysis of the obtained nonlinear fuzzy control system (with $k_p^2 = 1.5$) we shall use Corollary 2.7, resulting from Theorem 2.4. In order to do this, matrices $\mathbf{A}_{ij}$ (2.51) must be evaluated. The extended state vector in the considered example is: $x(k) = [y(k)\ y(k-1)\ u(k-1)]^T$, where dimensions $n = 2$ and $m = 1$ result from the form of the process and controller rule consequents. Thus, the matrices $A_{ij}$ are of dimension 3, and of the following general form

$$\mathbf{A}_{ij} = \begin{bmatrix} a_1^i - b_0^i c_1^j & a_2^i - b_0^i c_2^j & b_1^i + b_0^i d_1^j \\ 1 & 0 & 0 \\ -c_1^j & -c_2^j & d_1^j \end{bmatrix}, \quad i, j = 1, 2$$

Assuming, for a theoretical verification of stability conditions, the process modeled by (2.57), (2.58), *i.e.*, the same rule antecedents in the process and in the controller, it is enough to take into account matrices $\mathbf{A}_{11}$, $\mathbf{A}_{22}$ and $\overline{\mathbf{A}_{12}}$.

They have the following form

$$\mathbf{A}_{11} = \begin{bmatrix} -0.14 & 0.6 & 0.8 \\ 1 & 0 & 0 \\ -1.05 & 0.75 & 1 \end{bmatrix}$$

$$\mathbf{A}_{22} = \begin{bmatrix} -0.15 & 0.15 & 0.2 \\ 1 & 0 & 0 \\ -2.25 & 0.75 & 1 \end{bmatrix}$$

$$\overline{\mathbf{A}_{12}} = \begin{bmatrix} -0.5050 & 0.3750 & 0.5 \\ 1 & 0 & 0 \\ -1.65 & 0.75 & 1 \end{bmatrix}$$

Using the *LMI Toolbox* of the MATLAB® package it was checked that a system of linear matrix inequalities

$$\begin{aligned} (\mathbf{A}_{11})^T \mathbf{P} \mathbf{A}_{11} - \mathbf{P} &< 0 \\ (\mathbf{A}_{22})^T \mathbf{P} \mathbf{A}_{22} - \mathbf{P} &< 0 \\ (\overline{\mathbf{A}_{12}})^T \mathbf{P} \overline{\mathbf{A}_{12}} - \mathbf{P} &< 0 \\ -\mathbf{P} &< 0 \end{aligned}$$

has a solution

$$\mathbf{P} = 10^3 \begin{bmatrix} 5.4140 & -2.8931 & -3.0451 \\ -2.8931 & 2.2731 & 2.1858 \\ -3.0451 & 2.1858 & 2.5867 \end{bmatrix}$$

Thus, it follows from Corollary 2.7 (with $\mathbf{Q} = \mathbf{0}$) that sufficient conditions of stability of the considered nonlinear control system are satisfied. The conducted analysis applies to stability of a nominal system, there only remains to examine the robustness of the designed control system. It was checked that the system operates correctly even under quite significant deviations of its parameters. □

As mentioned previously, the form of the discrete TS fuzzy controller considered in this chapter also covers cases of predictive controllers of DMC and GPC type (in explicit, unconstrained versions). In Chapter 3 a design of nonlinear predictive controllers for the process from the example just shown above will be presented, including a GPC type controller.

## 2.3 Continuous-time TS Fuzzy Control

Until now, a discrete-time description of a process, and consequently, of controllers were assumed in our considerations. In practice, a discrete-time description is applied mainly to a situation when the sampling period in a control

system is not radically shorter than the dominant time constants of the process. For very small sampling periods we obtain, e.g.,, discrete step responses with a very large number of significant elements or difference equations of a high order, which unnecessarily makes the process of the design of control systems more difficult. On the other hand, as a result of the development of the microprocessor technology, controllers with a large computational power are widely available, which enables the use of small sampling periods even with more complex control algorithms. Thus, it is more common to use small sampling periods in direct control loops. Especially, because this approach enables a direct transfer from analog to digital technology – without the need to change the settings of the controllers.

In a situation when the sampling period in a control loop is small enough in relation to process dynamics, a fully authorized approach is to design a continuous-time control system, and then a digital realization of the obtained continuous-time controller by one of the emulation methods, see *e.g.*, [44, 52]. For that reason, in this section we shall be concerned with questions of the design and analysis of continuous (*i.e.*, continuous-time) TS fuzzy controllers. First of all, local control algorithms of such a controller will be defined as continuous. We should also mention that a verification of a control system by simulations can be conducted by formulating its description in a continuous time domain (*e.g.*, using the Simulink® program), or better yet, in a hybrid environment with a continuous realization of the process modeled by a set of differential equations integrated using an appropriate small step size and discrete-time controller realization.

Structures of continuous-time TS fuzzy algorithms are analogous to those for the discrete-time presented in the previous section. Therefore, considering design methods for continuous controllers and stability conditions of the obtained control systems we shall do it briefly, concentrating mainly on formulations and features distinguishing the two cases. Particularly, the first part of Section 2.2 was written in a general way, and thus refers to process modeling and to the design of TS fuzzy control algorithms, both in discrete and continuous versions.

### 2.3.1 Continuous TS Fuzzy State-feedback Controllers

Let us consider a dynamic process with the following state and control vectors

$$x(t) = [x_1(t)\ x_2(t)\ \cdots\ x_{n_x}(t)]^T$$

$$u(t) = [u_1(t)\ u_2(t)\ \cdots\ u_{n_u}(t)]^T$$

described by a TS fuzzy model with rules in the following form

$$\mathrm{R}_p^i: \quad \text{IF} \quad x_1(t) \text{ is } A_1^i \text{ and } x_2(t) \text{ is } A_2^i \text{ and } \cdots \text{ and } x_{n_x}(t) \text{ is } A_{n_x}^i$$
$$\text{THEN} \quad \dot{x}^i(t) = \mathbf{A}_i x(t) + \mathbf{B}_i u(t) \qquad (2.61)$$

where $A_j^i \in \mathbb{X}_j = \{X_{j1}, ..., X_{jr_j}\}$ are fuzzy sets of components $x_j$ of the state vector $x$ (see (2.4)), $j = 1, \ldots, n_x$, while $\mathbf{A}_i$ and $\mathbf{B}_i$ are state and control matrices of local linear models designed for individual fuzzy sub-domains of the model domain, $i = 1, ..., r$. When using a grid partition of the domain of the state variables we obtain a fuzzy system with a number of sub-domains (and rules) equal to $r = r_1 r_2 \cdots r_{n_x} = \prod_{j=1}^{n_x} r_j$.

For given values $x(t)$ and $u(t)$ the output of the fuzzy process model is calculated according to the standard formula, *i.e.*,

$$\dot{x}(t) = \frac{\sum_{i=1}^{r} w^i(t)[\mathbf{A}_i x(t) + \mathbf{B}_i u(t)]}{\sum_{l=1}^{r} w^l(t)} \tag{2.62}$$

where $w^i(t)$ are activation levels of individual rules (2.61),

$$w^i(t) = \prod_{j=1}^{n_x} \mu_{A_j^i}(x_j(t)).$$

We have taken a natural assumption that always $\sum_{i=1}^{r} w^i(k) > 0$, *i.e.*, for each value of the state variables from their domain the model is well defined – at least one rule is activated.

The model of an autonomous process (with $u(t) \equiv 0$) can be written in the form

$$\dot{x}(t) = \sum_{i=1}^{r} \tilde{w}^i(t) \mathbf{A}_i x(t) \tag{2.63}$$

where $\tilde{w}^i(t)$ denote normalized activation levels of the rules. For such a dynamic model, the following sufficient condition of asymptotic stability follows directly from the Lyapunov theorem.

**Theorem 2.8** *The equilibrium point of the dynamic system (2.63) is globally asymptotically stable if there exists a symmetric positive definite matrix* $\mathbf{P}$ *such that for the state matrix* $\mathbf{A}_i$ *of each local model the following inequality is satisfied*

$$\mathbf{A}_i^T \mathbf{P} + \mathbf{P} \mathbf{A}_i < \mathbf{0}, \quad i = 1, 2, ..., r \tag{2.64}$$

*Proof.* It can be conducted similarly as that of Theorem 2.1. We define a scalar function

$$V(x(t)) = x^T(t) \mathbf{P} x(t) \tag{2.65}$$

where $\mathbf{P}$ is a symmetric positive definite matrix. The function (2.65) from definition satisfies all prerequisites to be a Lyapunov function except for negativity of its derivative along trajectories of (2.63). The latter will be proven now. Namely,

$$\dot{V}(x(t)) = \dot{x}^T(t) \mathbf{P} x(t) + x^T(t) \mathbf{P} \dot{x}(t)$$

$$= x^T(t) \left[ \sum_{i=1}^{r} \tilde{w}^i(t) \mathbf{A}_i^T \mathbf{P} + \sum_{i=1}^{r} \tilde{w}^i(t) \mathbf{P} \mathbf{A}_i \right] x(t)$$

$$= \sum_{i=1}^{r} \tilde{w}^i(t) x^T(t) \left[ \mathbf{A}_i^T \mathbf{P} + \mathbf{P} \mathbf{A}_i \right] x(t)$$

Because

$$\tilde{w}^i(t) \geq 0 \quad \text{for every } i = 1, 2, ..., r$$

$$\sum_{i=1}^{r} \tilde{w}^i(t) > 0$$

then $\dot{V}(x(t)) < 0$ for $x(t) \neq 0$, if only (2.64) is satisfied. Thus, the function $V(x(t))$ is a Lyapunov function for the nonlinear dynamic system (2.63). □

It was shown in [141] that *a necessary condition for existence of a matrix* $\mathbf{P}$ *satisfying (2.64) is that each of the sums* $\mathbf{A}_i + \mathbf{A}_j$, $i, j = 1, 2, ..., r$ *(actually, each sum of matrices from among* $\mathbf{A}_i$, $i = 1, 2, ..., r$*) is a matrix of an asymptotically stable system* (all its eigenvalues have negative real parts). This results directly from addition of inequalities (2.64) for $i = i$ and $i = j$. Namely, if $\mathbf{P}$ is positive definite and $(\mathbf{A}_i + \mathbf{A}_j)^T \mathbf{P} + \mathbf{P}(\mathbf{A}_i + \mathbf{A}_j) < \mathbf{0}$, then $\mathbf{A}_i + \mathbf{A}_j$ is an asymptotically stable matrix. Thus, finding that any of the sums $\mathbf{A}_i + \mathbf{A}_j$ is not such a matrix indicates non-existence of the matrix $\mathbf{P}$ satisfying conditions of Theorem 2.8.

The existence of one common positive definite matrix $\mathbf{P}$ satisfying inequalities (2.64) is a sufficient condition of stability. However, necessary and sufficient conditions are as yet unknown. Conditions (2.64) are of global nature, they do not take into account fuzzy structure of the system (location and shapes of membership functions). That is why attempts were undertaken to derive weaker conditions by the use of a more structured Lyapunov function. The most natural construction would be the following

$$V(x(t)) = x^T(t) \mathbf{P}(x(t)) x(t) = x^T(t) [\sum_{i=1}^{r} \tilde{w}^i(t) \mathbf{P}_i] x(t) \tag{2.66}$$

where $\mathbf{P}_i$ are symmetric positive definite matrices. This function has nice properties, it is continuous, differentiable and positive definite. Unfortunately, to the best knowledge of the author the attempts to derive clear and reasonable conditions assuring its derivative is negative along the trajectory of the system (2.63) have not been successful. The reason is the dependence of $\mathbf{P}(x(t))$ on $x(t)$, which results in an additional term (stemming from the time derivative of $\mathbf{P}(x(t))$) in the expression for $\dot{V}(x(t))$, which makes it difficult to obtain constructive conditions assuring this derivative is negative along the system trajectories, see [23, 21, 10]. So far, the obtained conditions are either not constructive enough (a priori assumption that a norm of the derivative of

the state vector is appropriately bounded [21])), or are formulated in a too complex form, of unclear usefulness at actual stage of knowledge [10]. Therefore, they shall not be quoted here, referring an interested reader to the cited papers.

A theoretically more successful, although leading to a constrained and significantly more complex construction of the Lyapunov function, occurred to be the approach using a piecewise-quadratic Lyapunov function of the form

$$V(x(t)) = x^T(t)\mathbf{P}_l x(t), \quad x(t) \in S_l, \quad l = 1, ..., L \tag{2.67}$$

where $S_l$, $l = 1, ..., L$ constitute a partition of the process state-space into non-overlapping subsets of constant activation of all fuzzy rules (each rule is active or not within a subset) – called *constant activation cells* and constructed precisely in the same way as defined and explained in Section 2.2.1 for a discrete-time case.

However, the difficulty in a design of the function (2.67) as a Lyapunov function is the requirement of continuity when crossing boundaries of neighboring cells – the matrices $\mathbf{P}_l$ should be constructed in a way assuring this continuity. In [60] a design of such a continuous function was proposed, based on structuring matrices $\mathbf{P}_l$ as follows

$$\mathbf{P}_l = \mathbf{F}_l^T \mathbf{T} \mathbf{F}_l, \quad l = 1, ..., L \tag{2.68}$$

where $\mathbf{T}$ is a symmetric matrix and matrices $\mathbf{F}_l$ satisfy the following conditions

$$\mathbf{F}_l x = \mathbf{F}_m x \quad \text{for} \quad x \in S_l \cap S_m, \quad l, m \in L \tag{2.69}$$

In [60], see also [116], a way of construction of the matrices $\mathbf{F}_l$ for the case of trapezoidal membership functions was given, where the matrices are fully determined by the shape (by the corner points) of the membership functions. The stability of the autonomous fuzzy system

$$\dot{x}(t) = \sum_{i \in K(l)} \tilde{w}^i(t)[\mathbf{A}_i x(t)], \quad x(t) \in S(l), \quad l \in L \tag{2.70}$$

can then be proven provided a matrix $\mathbf{T}$ can be found such that matrices (2.68) are positive definite and the condition

$$\mathbf{A}_i^T \mathbf{P}_l + \mathbf{P}_l \mathbf{A}_i < \mathbf{0}, \quad i \in K(l), \quad l = 1, ..., L \tag{2.71}$$

holds, where $K(l)$ denotes a set containing indexes of all fuzzy rules with nonzero activation levels within $S_l$, $l = 1, ..., L$. For a detailed design and analysis the reader is referred to [60], as the design is difficult, leading to large sets of linear matrix inequalities (in fact, condition (2.71) in an augmented, more complex but slightly weakened form was there formulated). Moreover, a practical importance of the discussed stability conditions and their relation to the global result given by Theorem 2.8 seems to be not sufficiently understood yet (note that a global matrix $T$ is to be designed as well).

In [22] a completely different formulation of sufficient stability conditions for a dynamic system (2.63) was given.

**Theorem 2.9** *The equilibrium point of a fuzzy dynamic system (2.63) is globally asymptotically stable if*

$$\lambda_{\max}(\mathbf{A}_i + \mathbf{A}_i^T) < 0, \quad i = 1, 2, ..., r \tag{2.72}$$

*where* $\lambda_{\max}(\mathbf{A}_i + \mathbf{A}_i^T)$ *denotes the maximal eigenvalue of a symmetric matrix* $\mathbf{A}_i + \mathbf{A}_i^T$. □

The above theorem is quoted here without proof (see. [22]), as the given stability condition (2.72), although of a quite different nature than the one from Theorem 2.8 and formulated in terms of local matrices, is difficult to interpret and generally turns out to be very conservative, thus of not much use.

For each local linear dynamic model from the rule consequents (2.61), *i.e.*, for each sub-domain of a TS fuzzy model, a linear state-feedback controller can be designed, using a classical method. This way the rules of a TS fuzzy controller are obtained, in the form

$$\begin{aligned} \mathrm{R}_c^j: \quad \text{IF} \quad & x_1(t) \text{ is } A_1^j \text{ and } x_2(t) \text{ is } A_2^j \text{ and } \cdots \text{ and } x_{n_x}(t) \text{ is } A_{n_x}^j \\ & \text{THEN} \qquad u^j(t) = -\mathbf{F}_j x(t) \end{aligned} \tag{2.73}$$

where $\mathbf{F}_i$ are matrices of state-feedback coefficients, $j = 1, 2, ..., r$. A nonlinear TS fuzzy controller is described by

$$u(t) = -\sum_{j=1}^{r} \tilde{w}^j(t) \mathbf{F}_j x(t) \tag{2.74}$$

where normalized levels of rule activation are the same as in the process model (2.61) used for the design of the controller.

To represent a process in the closed-loop control system, it is generally possible to use a TS fuzzy model different than the one used for the design of the fuzzy controller, *i.e.*, with a different number and/or shape of the rules (analogously as it was discussed in Section 2.2.1). The number of rules of this model is denoted by $ro$, and their activation levels by $w_o^i(k)$, $i = 1, ..., ro$. Substituting now the description of the controller (2.74) into the fuzzy process model (2.62) – but with $ro$ rules with activation levels $w_o^i(k)$ – we obtain a formula describing the closed-loop control system

$$\begin{aligned} \dot{x}(t) &= \frac{\sum_{i=1}^{ro} \sum_{j=1}^{r} w_o^i(t) w^j(t) (\mathbf{A}_i - \mathbf{B}_i \mathbf{F}_j) x(t)}{\sum_{l=1}^{ro} \sum_{p=1}^{r} w_o^l(t) w^p(t)} \\ &= \sum_{i=1}^{ro} \sum_{j=1}^{r} \tilde{w}_o^i(t) \tilde{w}^j(t) (\mathbf{A}_i - \mathbf{B}_i \mathbf{F}_j) x(t) \end{aligned} \tag{2.75}$$

Of course, if a model used to represent the process in the closed-loop control system is different than the model (2.61)-(2.62) which was used for the design

of the controller, then its matrices are, generally, also different. However, in the control system description (2.75) only the model with matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ representing the process in the closed loop occurs.

In order to examine stability of a dynamic system described by (2.75) it is possible to directly apply Theorem 2.8 or Theorem 2.9, only matrices $\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_j$, $i = 1, ..., ro$, $j = 1, ..., r$ instead of the matrices $\mathbf{A}_i$, $i = 1, ..., r$ must be considered. However, if the number of rules and its antecedents used to describe a process in the closed-loop control system are the same as in the fuzzy description of the controller (thus $ro = r$ and $w_o^i(k) = w^i(k)$), then (2.75) can be simplified – in an analogous way as it was done in Section 2.2.1 with the formula (2.25) for systems with discrete time. Namely, applying equalities $w^i(k)w^j(k) = w^j(k)w^i(k)$, $i, j = 1, 2, ..., r$, (2.75) can be transformed to the form

$$\dot{x}(t) = \sum_{i=1}^{r} \tilde{w}^i(t)\tilde{w}^i(t)(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_i)x(t) + 2 \sum_{i,j=1,i<j}^{r} \tilde{w}^i(t)\tilde{w}^j(t)\mathbf{D}_{ij}x(t) \quad (2.76)$$

where

$$\mathbf{D}_{ij} = \frac{1}{2}[(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_j) + (\mathbf{A}_j - \mathbf{B}_j\mathbf{F}_i)], \quad i < j,\ i, j = 1, 2, ..., r \quad (2.77)$$

Sufficient stability conditions can now be formulated in the following form, see [132].

**Theorem 2.10** *Let the maximum number of rules activated simultaneously in the control system (2.76) described by the rules of the process model (2.61) and controller (2.73) be no higher than $s$, $1 < s \leq r$. Then the equilibrium point of this system is globally asymptotically stable, if there exist a positive definite matrix $\mathbf{P}$ and a positive semi-definite matrix $\mathbf{Q}$ satisfying the following conditions*

$$(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_i)^T\mathbf{P} + \mathbf{P}(\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_i) + (s-1)\mathbf{Q} < 0, \quad i = 1, 2, ..., r$$
$$\mathbf{D}_{ij}^T\mathbf{P} + \mathbf{P}\mathbf{D}_{ij} - \mathbf{Q} < 0, \quad i < j,\ i, j = 1, 2, ..., r$$

*except for pairs $(i, j)$ for which always $w^i(t)w^j(t) = 0$.* □

In particular, for $\mathbf{Q} = \mathbf{0}$ we obtain a slightly sharper formulation of the stability conditions which can be used for initial stability verification. In [140] a further, rather slight weakening of the assumptions of Theorem 2.10 was obtained, using the same Lyapunov function $V(x(t)) = x^T(t)\mathbf{P}x(t)$. The obtained conditions in a form of a set of LMIs (Linear Matrix Inequalities) are however much more complex.

It is also possible to try to use Theorem 2.9 for the case of a closed-loop control system, then we obtain

**Corollary 2.11** *The equilibrium point of a dynamic system (2.76) is globally asymptotically stable if*

$$\lambda_{\max}([\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_j] + [\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_j]^T) < 0, \quad i, j = 1, 2, ..., r$$

*except for pairs* $(i, j)$ *for which always* $w^i(t)w^j(t) = 0$. □

If it is necessary to reconstruct the state vector, it is possible to use a nonlinear state observer in the form of a TS fuzzy system with a rule structure and antecedents identical to those of the process model and the controller, but with functional consequents of the rules which are classical state observer equations, see [132].

**Example 2.7**
The task of stabilization of an inverted pendulum attached to a cart, presented in Fig. 2.27, will be considered. Equations describing process dynamics are as follows (see *e.g.*, [106, 141])

$$\dot{x}_1 = x_2 \tag{2.78a}$$

$$\dot{x}_2 = \frac{g \sin x_1 - aml(x_2)^2 \sin(2x_1)/2 - a\cos(x_1)u}{4l/3 - aml\cos^2 x_1} \tag{2.78b}$$

where $x_1$ denotes the angle between actual position of the pendulum arm and vertical direction ($0\,[rad]$ corresponds to the vertical position of the pendulum), hence $\dot{x}_1 = x_2$ denotes angular velocity. Masses of the pendulum and the cart are denoted by $M$ and $m$, respectively, $2l$ is its length and $a = 1/(M+m)$. A force imposed on the cart is the control variable $u$, $g = 9.81\,[m/sec^2]$ is the gravity acceleration. The following numerical values are assumed: m = $2\,[kg]$, M = $8\,[kg]$, 2l = $1\,[m]$ (as in [150, 141], where the control of the presented process is also investigated).

It is known that stabilization of the pendulum in the vertical position by a linear controller works correctly for a limited range of angular deviations. We shall design a simple TS fuzzy controller consisting of only two rules, operating more efficiently and for a wider range of deviations. The first operating region will be around the zero position, denoted by $X_1$ and defined by a two-sided sigmoidal membership function in the following form

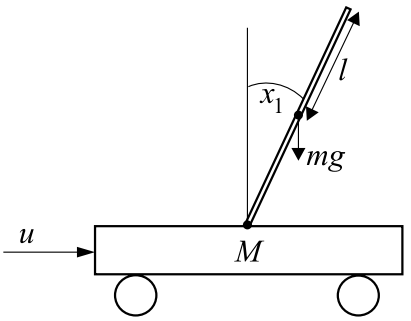


**Fig. 2.27.** Inverted pendulum on a cart

$$\text{sigmf1}(x) = \frac{1}{1+\exp[-15(x_1-\pi/8)]} - \frac{1}{1+\exp[-15(x_1+\pi/8)]} \tag{2.79}$$

and the second region will be around the positions $x_1 = \pm\pi/4$, denoted by $X_2$ and defined by the membership function

$$\text{sigmf2}(x) = 1 - \text{sigmf1}(x) \tag{2.80}$$

In order to design a fuzzy controller we shall make an approximation of the nonlinear description (2.78a)-(2.78b) by a TS fuzzy model with the rules

$$\text{R}_p^1: \text{ IF } \ x_1(t) \text{ is } X_1 \ \text{ THEN } \ \dot{x}^1(t) = \mathbf{A}_1 x(t) + \mathbf{B}_1 u(t) \tag{2.81a}$$

$$\text{R}_p^2: \text{ IF } \ x_1(t) \text{ is } X_2 \ \text{ THEN } \ \dot{x}^2(t) = \mathbf{A}_2 x(t) + \mathbf{B}_2 u(t) \tag{2.81b}$$

where matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ will now be designed.

First linear model will be designed for the first region around the equilibrium point [0 0]. Performing a standard linearization we obtain

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 \\ \frac{g}{4l/3-aml} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 17.3118 & 0 \end{bmatrix}$$

$$\mathbf{B}_1 = \begin{bmatrix} 0 \\ \frac{-a}{4l/3-aml} \end{bmatrix} = \begin{bmatrix} 0 \\ -0.1765 \end{bmatrix}$$

However, performing a standard linearization (based on Taylor series expansion) of the original nonlinear model (2.78a)-(2.78b) at a non-zero point $[\pi/4,\ 0]$, we would obtain an affine model, not a linear one which is used for construction of a local state-feedback matrix. A possible solution is to use a slightly different way of linearization, as proposed in [141]. We shall proceed along these lines, using the general structure of our nonlinear model, namely

$$\dot{x}(t) = \mathbf{f}(x(t)) + \mathbf{G}(x(t))u(t)$$

It is required that the linear model $\dot{x}(t) = \mathbf{A}_2 x(t) + \mathbf{B}_2 u(t)$ approximates a nonlinear description, in the vicinity of the point $x_0 = [\pi/4 \ \ 0]$, in such a way that

$$\mathbf{f}(x) + \mathbf{G}(x)u \approx \mathbf{A}_2 x + \mathbf{B}_2 u$$

and

$$\mathbf{f}(x_0) + \mathbf{G}(x_0)u = \mathbf{A}_2 x_0 + \mathbf{B}_2 u, \ u \in \mathbb{R}$$

It follows directly from these conditions that

$$\mathbf{B}_2 = \mathbf{G}(x_0)$$

Thus, what remains to be fulfilled is

$$\mathbf{f}(x) \approx \mathbf{A}_2 x \ \text{ and } \ \mathbf{f}(x_0) = \mathbf{A}_2 x_0$$

Let us denote by $\mathbf{a}_{2j}$ rows of the matrices $\mathbf{A}_2$, $j = 1, 2$. It was shown in [141] that to satisfy the conditions formulated above the rows $\mathbf{a}_{2j}$ should result from the solution of the following optimization problem

$$\min_{\mathbf{a}_{2j}} \{0.5 \|\nabla f_j(x_0) - \mathbf{a}_{2j}\|_2^2\}$$
$$\text{subj. to: } \mathbf{a}_{2j}^T x_0 = f_j(x_0)$$

where $\mathbf{f} = [f_1 \ f_2]^T$. An analytical solution of the formulated quadratic optimization problem is given by the formula

$$\mathbf{a}_{2j} = \nabla f_j(x_0) + \frac{f_j(x_0) - x_0^T \nabla f_j(x_0)}{\|x_0\|^2} x_0, \quad x_0 \neq 0$$

Using the above reasoning we obtain the following matrices $\mathbf{A}_2$ and $\mathbf{B}_2$:

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 1 \\ \frac{4g \sin(\pi/4)}{\pi[4l/3 - aml \cos^2(\pi/4)]} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 14.3223 & 0 \end{bmatrix}$$
$$\mathbf{B}_2 = \begin{bmatrix} 0 \\ \frac{-a \cos(\pi/4)}{4l/3 - aml \cos^2(\pi/4)} \end{bmatrix} = \begin{bmatrix} 0 \\ -0.1147 \end{bmatrix}$$

A TS fuzzy controller will be described by the rules

$$\mathrm{R}_c^1: \quad \text{IF} \quad x_1(t) \text{ is } X_1 \quad \text{THEN} \quad u^1(t) = -\mathbf{F}_1 x(t)$$
$$\mathrm{R}_c^2: \quad \text{IF} \quad x_1(t) \text{ is } X_2 \quad \text{THEN} \quad u^2(t) = -\mathbf{F}_2 x(t)$$

and the final inference formula

$$u(t) = w^1(t)\, u^1(t) + w^2(t)\, u^2(t)$$

where $w^1(t)$ and $w^2(t)$ are activation levels of the rules (let us note that it follows from the construction of membership functions (2.79) and (2.80) that $w^1(t) + w^2(t) = 1$). Matrices $\mathbf{F}_1$ and $\mathbf{F}_2$ will be obtained in a standard way, assuming appropriate placement of eigenvalues of matrices $\mathbf{A}_i - \mathbf{B}_i \mathbf{F}_i$, $i = 1, 2$. If we require that for both linear models these values should be $-2$ and $-2.5$, then using the command *place* from the *Control Systems Toolbox* of the MATLAB® package, we obtain

$$\mathbf{F}_1 = [-126.4125 \ -25.4958]$$
$$\mathbf{F}_2 = [-168.4595 \ -39.2328]$$

For simulations of the closed-loop control system the original nonlinear process model (2.78a)-(2.78b) was used and the designed TS fuzzy controller, which generated the control signal according to

$$u(t) = -w^1(t)\, \mathbf{F}_1\, x(t) - w^2(t)\, \mathbf{F}_2\, x(t)$$

Selected trajectories of state variables $x(t)$ for initial conditions $x_1(0)$ equal to 0.25, 0.5 and 1.0 $[rad]$ and $x_2(0) = 0\,[rad/sec]$ are presented in Fig. 2.28. For a comparison, Fig. 2.29 presents trajectories in the control system with the fuzzy controller and a linear controller designed for the first area ($u(t) = -\mathbf{F}_1 x(t)$), for initial state $x_1(0) = 0.5$, $x_2(0) = 0$. Generally, the further the pendulum position from a vertical one the faster the operation of the nonlinear fuzzy controller. Moreover, it regulates the pendulum to a vertical position from a wider range of initial states. It still operates starting from $x_1(0) = 1.0$, while the linear controller is already unstable for $x_1(0) = 0.85$.

In order to check the quality of fuzzy modeling of the considered strongly nonlinear process, simulations were also conducted for the control system with the process represented by the TS fuzzy model given by (2.81a)-(2.81b). Trajectories obtained in this control system are shown by dashed lines in Fig. 2.30, where trajectories corresponding to the control system with the original nonlinear process are also shown (solid lines). In the range of smaller angular deviations the coincidence of these two types of trajectories is very good (the trajectories for $x_1(0) = 0.25$ are almost undistinguishable).

The above analysis is interesting since we can apply theoretical stability criteria when both the process and the controller are described by fuzzy models in the control system loop. Applying Theorem 2.10 with $\mathbf{Q} = \mathbf{0}$, it can be concluded that the sufficient stability condition is satisfied, with the matrix

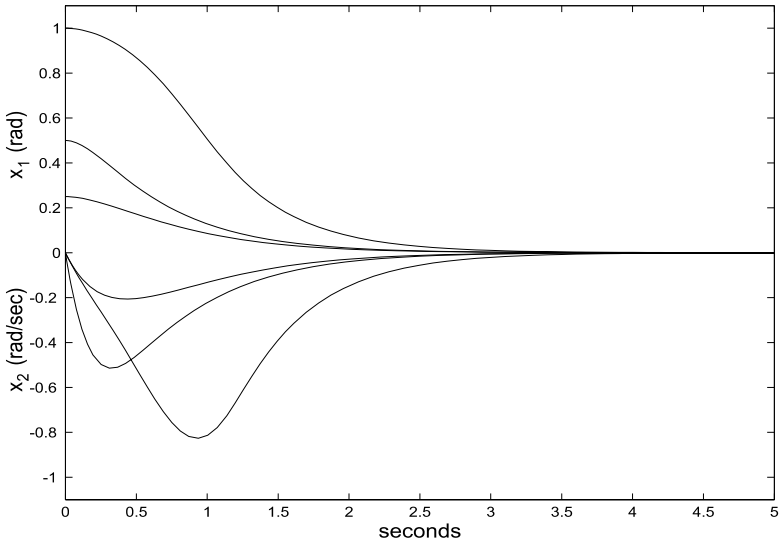


**Fig. 2.28.** Trajectories in the control system with the TS fuzzy controller, for various initial conditions

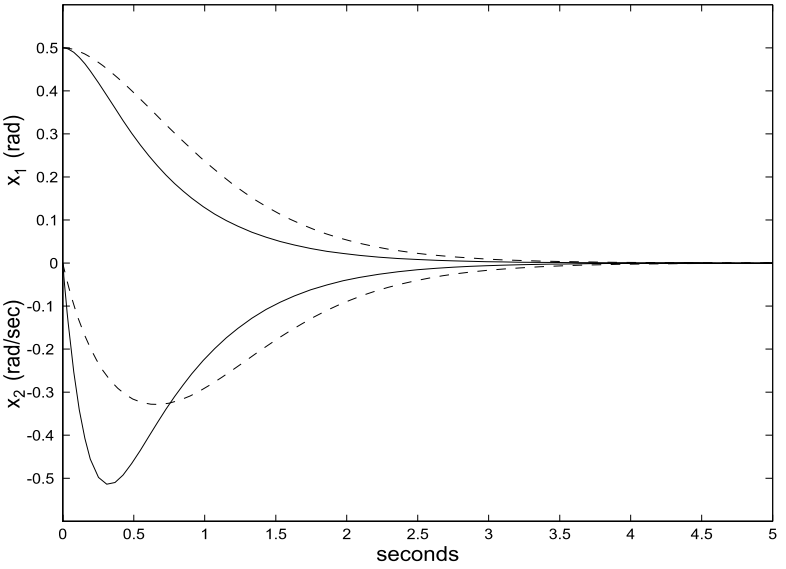


**Fig. 2.29.** Trajectories in the control system with the TS fuzzy controller (solid line) and with the linear controller (dashed line)

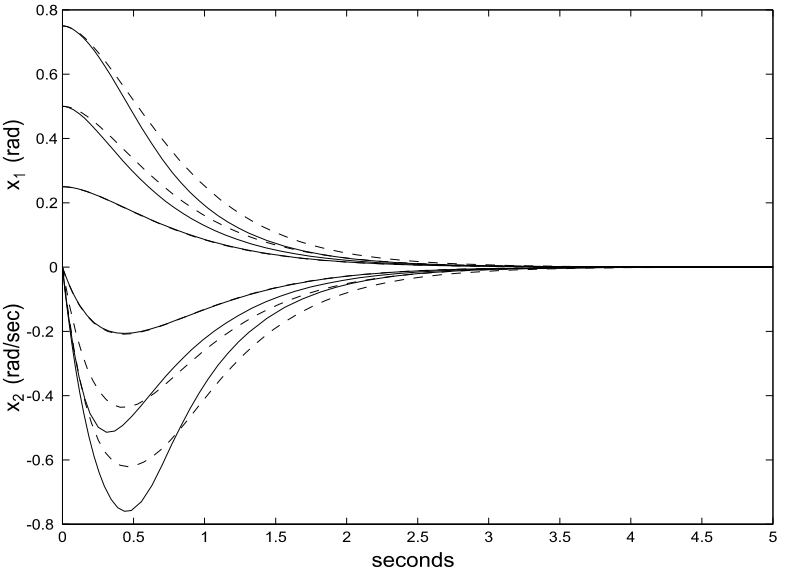


**Fig. 2.30.** Trajectories in control systems with the TS fuzzy controller and the original nonlinear process model (solid lines), and with the TS fuzzy process model (dashed lines)

$$\mathbf{P} = \begin{bmatrix} 1.2529 & 0.1025 \\ 0.1025 & 0.1461 \end{bmatrix}$$

It was also investigated that the control system is stable with different (perturbed) state-feedback matrices $\mathbf{F}_i$, corresponding to eigenvalues of the matrices $\mathbf{A}_i - \mathbf{B}_i\mathbf{F}_i$ from the range $-1$ to $-4$.

If credibility of this sort of analysis should be increased, it would be necessary to construct a more precise TS fuzzy model, with a larger number of rules.

Applying Corollary 2.11 to the stability analysis of the considered control system does not lead to a constructive result, unfortunately – the matrices inspected have positive eigenvalues. This confirms the author's opinion about a conservative character of assumptions of Theorem 2.9 and about limited and unclear, so far, range of usefulness of this theorem. □

### 2.3.2 Continuous TS Fuzzy Output-feedback Controllers

In practice of industrial process control the most important role is played by controllers with feedback from process outputs and with settings based on an approximate process model, as a state vector is usually not available for current measurements and we rarely have a precise model of the process dynamics at our disposal. Especially, in direct control loops PID controllers are still dominant. As mentioned in the introduction to this chapter, in situations of frequent changes of operating points and significant nonlinearities, the PID linear controllers do not always ensure appropriate quality of the control. One of the most effective solutions then is to apply a TS fuzzy control with feedback from measured process outputs. In Section 2.2.2 problems of the design and analysis of discrete-time TS fuzzy output-feedback controllers were presented. In this section we shall deal with continuous TS fuzzy output-feedback controllers, designed for a continuous-time process description. The obtained continuous control algorithm is then transformed into a discrete one by one of the emulation methods and implemented in a digital controller, with the sampling period sufficiently small in relation to the dynamics of the controlled process.

The considerations will be limited to the case of a PID controller which is the most important one in practice. The methodology of the design will be, however, of a general type, applicable to any linear dynamic control algorithm.

The basic question when starting a design of a continuous TS fuzzy controller, analogously as it was in the case of a discrete controller, is the choice of variables for antecedents of the controller rules and a design of fuzzy sets for these variables. That is, a partitioning of domains of these variables into mutually overlapping fuzzy sets defining altogether overlapping sub-domains (multivariable fuzzy sets) of the whole process domain. This should be done in such a way that in each of these sub-domains the process can be controlled by an appropriately selected linear controller, designed for an operating point

central for the sub-domain. Process operating points are related to certain values of process states, therefore the variables occurring in the rule antecedents of a TS fuzzy model will be, first of all, state variables describing nonlinear behavior of the process. These will be mainly the controlled variable and its derivatives. The variables of the rule antecedents will be denoted by $x_1(t), ..., x_n(t)$, and fuzzy sets describing the partitioning of the domain of each of them will be denoted by $A_j^i$, $j = 1, ..., n_x$.

Having defined the fuzzy sub-domains of the process operational space and its central points (process operating points), in each of them a local continuous PID controller is tuned, *e.g.*, using one of the known, standard tuning methods. In this way a *continuous TS fuzzy PID* (FPID) controller has been obtained, defined by the rules

$$\begin{aligned} \mathrm{R}_c^j: \quad & \text{IF} \quad x_1(t) \text{ is } A_1^j \text{ and } x_2(t) \text{ is } A_2^j \text{ and } \cdots \text{ and } x_{n_x}(t) \text{ is } A_{n_x}^j \\ & \text{THEN } u^j(t) = k_P^j \left[ e(t) + \frac{1}{T_I^j} \int_0^t e(\tau) d\tau + T_D^j \frac{de(t)}{dt} \right] \end{aligned} \tag{2.82}$$

where $j = 1, \ldots, r$ indexes rules (and at the same time local operating subdomains), $e(t) = y^{sp}(t) - y(t)$ is the control error at time $t$, whereas settings of the local PID controllers: gain $k_P^j$, integral time constant $T_I^j$ and derivative time constant $T_D^j$ are coefficients of the functions of the rule consequents.

The output signal of the controller takes a standard form, for TS fuzzy structures,

$$u(t) = \frac{\sum_{j=1}^r w^j(t)\, u^j(t)}{\sum_{l=1}^r w^l(t)} = \sum_{j=1}^r \tilde{w}^j(t)\, u^j(t) \tag{2.83}$$

where $w^j(t)$ are activation levels of individual rules (2.82) at time $t$,

$$w^j(t) = \prod_{p=1}^{n_x} \mu_{A_p^j}(x_p(t)) \tag{2.84}$$

and $\tilde{w}^j(t)$ are their normalized values.

Implementing (2.82) and (2.83) in a simulation environment, such as Simulink®, we obtain a realization of a continuous TS fuzzy PID controller. In order to perform simulations of the entire system it is also necessary to have a nonlinear model of the dynamic process itself, formulated in a form accepted by the employed simulation environment. If we want to support the simulation experiments by a theoretical analysis of stability of the control system, then we currently have methods of such an analysis using process and controller models formulated in the form of state equations. State equations describe the dynamics directly in the time domain, a domain which is natural for formulation of nonlinear fuzzy models and for analysis of stability of their equilibrium points. The basic tool for this analysis is the Lyapunov method, see *e.g.*, [61, 148, 132, 141, 21, 140], but almost all publications concern

systems with controllers with linear state-feedback (only in [23] an output-feedback controller is considered). In the sequel, we shall present an analysis of stability of a continuous control system with a TS fuzzy controller, limiting the derivation of appropriate equations to the case of a PI type controller.

To describe the dynamics of a closed-loop control system it is necessary to have a TS fuzzy process model, using a model with the same rule antecedents as in controller rules (2.82) will simplify the analysis. Obviously, it can also be a fuzzy model with a different number of rules and different form of their antecedents, in general. Similarly as in the description of control systems in previous sections, we shall denote the number of process model rules by $ro \neq r$. A functional consequent of each rule is a local linear process model given in the form of the following state equations

$$\dot{x}^i(t) = \mathbf{A}_i x(t) + \mathbf{B}_i u(t)$$
$$y^i(t) = \mathbf{C} x^i(t)$$

$i = 1, 2, ..., ro$. Denoting normalized activation levels of the process model rules by $\tilde{w}_o^i(k)$ we obtain equations of the entire nonlinear process model in the form

$$\dot{x}(t) = \sum_{i=1}^{ro} \tilde{w}_o^i(t)[\mathbf{A}_i x(t) + \mathbf{B}_i u(t)] \tag{2.85}$$
$$y(t) = \mathbf{C} x(t) \tag{2.86}$$

We shall add to these equations dynamics of the integrator of the control error

$$\dot{q}(t) = y^{sp} - \mathbf{C} x(t)$$

The rules of a PI controller have the following form

$$\mathrm{R}_c^j: \text{ IF } \quad x_1(t) \text{ is } A_1^j \text{ and } x_2(t) \text{ is } A_2^j \text{ and } \cdots \text{ and } x_{n_x}(t) \text{ is } A_{n_x}^j$$
$$\text{THEN } \quad u^j(t) = k_P^j e(t) + \frac{k_P^j}{T_I^j} q(t) \tag{2.87}$$

$j = 1, 2, ..., r$, where $e(t) = y^{sp}(t) - \mathbf{C} x(t)$. The controller output is given by

$$\begin{aligned} u(t) &= \sum_{j=1}^{r} \tilde{w}^j(t)[k_P^j e(t) + \frac{k_P^j}{T_I^j} q(t)] \\ &= -\sum_{j=1}^{r} \tilde{w}^j(t) k_P^j \mathbf{C} x(t) + \sum_{j=1}^{r} \tilde{w}^j(t) \frac{k_P^j}{T_I^j} q(t) + \sum_{j=1}^{r} \tilde{w}^j(t) k_P^j y^{sp}(t) \end{aligned} \tag{2.88}$$

Substituting this equation into (2.85) we obtain

$$\dot{x}(t) = \sum_{i=1}^{ro} \tilde{w}_o^i(t) \left\{ [\mathbf{A}_i - \mathbf{B}_i \sum_{j=1}^{r} \tilde{w}^j(t) k_P^j \mathbf{C}] x(t) \right.$$
$$\left. + \mathbf{B}_i \sum_{j=1}^{r} \tilde{w}^j(t) \frac{k_P^j}{T_I^j} q(t) + \mathbf{B}_i \sum_{j=1}^{r} \tilde{w}^j(t) k_P^j y^{sp}(t) \right\}$$
$$= \sum_{i=1}^{ro} \sum_{j=1}^{r} \tilde{w}_o^i(t) \tilde{w}^j(t) \left\{ [\mathbf{A}_i - \mathbf{B}_i k_P^j \mathbf{C}] x(t) + \mathbf{B}_i \frac{k_P^j}{T_I^j} q(t) + \mathbf{B}_i k_P^j y^{sp}(t) \right\}$$

If we introduce now the following extended state vector

$$v(t) = [x(t)^T \; q(t)]^T \tag{2.89}$$

then, using the equalities $\sum_{i=1}^{ro} \tilde{w}_o^i(t) = \sum_{j=1}^{r} \tilde{w}^j(t) = 1$, we can write equations of the dynamics of the closed-loop control system in the following form

$$\dot{v}(t) = \sum_{i=1}^{ro} \sum_{j=1}^{r} \tilde{w}_o^i(t) \tilde{w}^j(t) \left\{ \begin{bmatrix} \mathbf{A}_i - \mathbf{B}_i k_P^j \mathbf{C} & \mathbf{B}_i \frac{k_P^j}{T_I^j} \\ -\mathbf{C} & 0 \end{bmatrix} v(t) + \begin{bmatrix} \mathbf{B}_i k_P^j \\ 1 \end{bmatrix} y^{sp}(t) \right\}$$
$$= \sum_{i=1}^{ro} \sum_{j=1}^{r} \tilde{w}_o^i(t) \tilde{w}^j(t) \left\{ [\tilde{\mathbf{A}}_i - \mathbf{G}_{ij}] v(t) \right\} + \sum_{i=1}^{ro} \sum_{j=1}^{r} \tilde{w}_o^i(t) \tilde{w}^j(t) \tilde{\mathbf{B}}_{ij} y^{sp}(t)$$

where

$$\tilde{\mathbf{A}}_i = \begin{bmatrix} \mathbf{A}_i & \mathbf{0} \\ -\mathbf{C} & 0 \end{bmatrix}, \; \mathbf{G}_{ij} = \begin{bmatrix} \mathbf{B}_i k_P^j \mathbf{C} & -\mathbf{B}_i \frac{k_P^j}{T_I^j} \\ \mathbf{0} & 0 \end{bmatrix}, \; \tilde{\mathbf{B}}_{ij} = \begin{bmatrix} \mathbf{B}_i k_P^j \\ 1 \end{bmatrix}$$

Without loos of generality we can assume that $y^{sp}(t) = 0$. Then equations of the dynamics of our system take the following form

$$\dot{v}(t) = \sum_{i=1}^{ro} \sum_{j=1}^{r} \tilde{w}_o^i(t) \tilde{w}^j(t) [\tilde{\mathbf{A}}_i - \mathbf{G}_{ij}] v(t) \tag{2.90}$$

which is identical to the dynamics of the state-feedback control system considered in the previous section described by (2.75), where matrices $\mathbf{A}_i$ and $\mathbf{B}_i \mathbf{F}_j$ in (2.75) correspond to matrices $\tilde{\mathbf{A}}_i$ and $\mathbf{G}_{ij}$ in (2.90). To examine stability of the dynamic system described by (2.90) it is possible to apply directly Theorem 2.8 or Theorem 2.9, only matrices $\tilde{\mathbf{A}}_i - \mathbf{G}_{ij}$, $i = 1, ..., ro$, $j = 1, ..., r$ should be considered instead of matrices $\mathbf{A}_i$, $i = 1, ..., r$.

Moreover, if we assume that the number and antecedents of the rules in the process model and in the controller are the same, then defining the matrix

$$\tilde{\mathbf{D}}_{ij} = \frac{1}{2}[(\tilde{\mathbf{A}}_i - \mathbf{G}_{ij}) + (\tilde{\mathbf{A}}_j - \mathbf{G}_{ji})], \quad i < j, \; i, j = 1, 2, ..., r \tag{2.91}$$

analogous to the matrix $\mathbf{D}_{ij}$ (2.77), we obtain (2.90) in a transformed form

$$\dot{v}(t) = \sum_{i=1}^{r} w^i(t)w^i(t)(\tilde{\mathbf{A}}_i - \mathbf{G}_{ii})v(t) + 2\sum_{i,j=1,i<j}^{r} w^i(t)w^j(t)\tilde{\mathbf{D}}_{ij}v(t) \quad (2.92)$$

corresponding to (2.76). Thus, to analyze stability of the dynamic system (2.92) it is possible to apply directly Theorem 2.10 (also Corollary 2.11), only matrices $\mathbf{A}_i$, $\mathbf{B}_i\mathbf{F}_i$ and $\mathbf{D}_{ij}$ occurring there should be replaced with matrices $\tilde{\mathbf{A}}_i$, $\mathbf{G}_{ii}$ and $\tilde{\mathbf{D}}_{ij}$.

The method of an analysis of a continuous control system with an output-feedback controller was presented above by way of the example of a PI controller. In a similar way, we can treat another cases with linear dynamic output-feedback controllers.

Let us note that due to linearity of the rule consequents (2.82) the dependence (2.83) can be written in the following form:

$$\begin{aligned} u(t) &= \sum_{j=1}^{r} \tilde{w}^j(t) \cdot k_P^j \left[ e(t) + \frac{1}{T_I^j}\int_0^t e(\tau)d\tau + T_D^j \frac{de(t)}{dt} \right] \\ &= \sum_{j=1}^{r} \tilde{w}^j(t)k_P^j \cdot e(t) + \sum_{j=1}^{r} \tilde{w}^j(t)\frac{k_P^j}{T_I^j} \cdot \int_0^t e(\tau)d\tau + \sum_{j=1}^{r} \tilde{w}^j(t)k_P^j T_D^j \cdot \frac{de(t)}{dt} \end{aligned} \quad (2.93)$$

Therefore, the considered fuzzy controller can be treated as a nonlinear PID controller, with the parameters varying in a nonlinear way, continuously adapting to the changes in the process operating point, because $\tilde{w}^j(t) = \tilde{w}^j(x(t))$, see (2.84). This feature, together with the relative simplicity and clarity of the design method, has been decisive in the success of TS fuzzy controllers and especially of TS fuzzy PID (TS-FPID) controllers. The above interpretation indicates also that in the case of an identical structure of linear consequents in all rules of a TS fuzzy controller, as in the considered case of the TS-FPID controller, on-line changing its settings can be treated as applying a kind of *gain scheduling* technique, see *e.g.*, [2]. However, it is performed in a completely different way: not by using a nonlinear function approximating an inverse of the process nonlinearity, but by using a more general and simpler mechanism of fuzzy reasoning.

**Example 2.8**

A nonlinear control of pH value of a mixture in a continuous-flow reactor presented in Fig. 2.31 will be considered.

Dynamics of the reactor is described by the following equations:

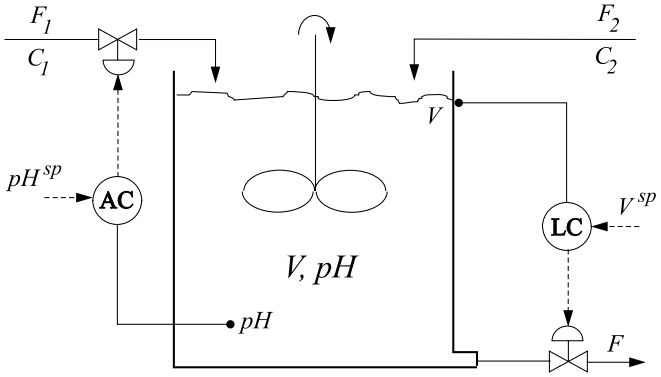


**Fig. 2.31.** Structure of pH and level control in a continuous-stirred tank reactor, Example 2.8

$$\frac{dV\xi}{dt} = F_1C_1 - (F_1 + F_2)\xi \tag{2.94}$$

$$\frac{dV\psi}{dt} = F_2C_2 - (F_1 + F_2)\psi \tag{2.95}$$

$$\frac{dV}{dt} = F_1 + F_2 - F \tag{2.96}$$

$$\left[H^+\right]^3 + (K_a + \psi)\left[H^+\right]^2 + (K_a(\psi - \xi) - K_w)\left[H^+\right] - K_aK_w = 0 \tag{2.97}$$

where

$\xi \sim [HAC] + [AC^-]$,

$\psi \sim [Na^+]$,

$pH = -\log_{10}[H^+]$,

$C_1 = 0.32\,[\frac{mol}{l}]$ – acid concentration in the inflow $F_1$,

$C_2 = 0.05005\,[\frac{mol}{l}]$ – acid concentration in the inflow $F_2$,

$V = 1000\,[l]$ – volume of the mixture,

$K_a$ i $K_w$ – equilibrium constants of acid and water,

$K_a = 1.8 \times 10^{-5}$, $K_w = 1.0 \times 10^{-14}$,

and the following nominal values of the inflow rates were assumed:

$F_1(0) = 81\,[\frac{l}{min}]$, $F_2(0) = 512\,[\frac{l}{min}]$ .

There are two feedback control loops in the system, see Fig 2.31:

- Control of the volume $V$ (level control), where the outflow rate $F$ is the manipulated variable,
- Concentration of pH, where the inflow rate $F_1$ is the manipulated variable,

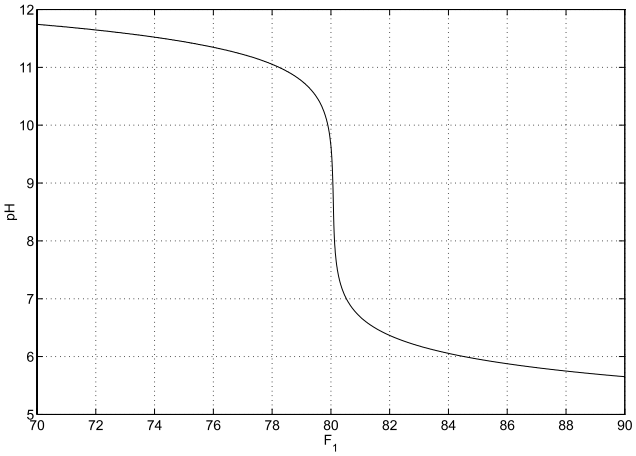


**Fig. 2.32.** Static dependence of $pH$ versus $F_1$

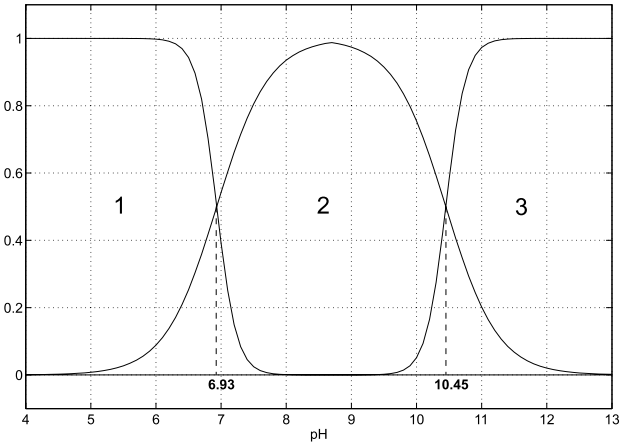


**Fig. 2.33.** Membership functions of the fuzzy $pH$ controller

while the inflow rate $F_2$ is a disturbance (uncontrolled process input).

The first feedback control loop is standard and a typical linear controller easily ensures a sufficient control quality. However, controlling the pH value turns out to be very difficult, which results from a strongly nonlinear static dependence of pH concentration versus the inflow $F_1$, as it is shown in Fig. 2.32. Moreover, the operating point of interest ($pH = 8.7$) is located in the area of strongest nonlinearity, which makes it impossible to select one PI controller for the entire domain of the control. A controller operating correctly in the

area of high process gain ($pH = 7.5 \div 10$) operates unacceptably slowly in both areas of smaller gain, *i.e.*, for small and large values of $pH$. On the other hand, a controller operating better (faster) in these areas becomes unstable in the middle area of high gain. Therefore, a nonlinear TS fuzzy controller was designed. The only variable present in the antecedents of the controller rules is the $pH$ value. A partition of the $pH$ domain into three fuzzy sets was assumed, with sigmoidal membership functions presented in Fig. 2.33. The PI controllers were designed for each of the fuzzy sets, with settings given in Table 2.1.

**Table 2.1.** Parameters of local PI controllers

| controller$j$ | $k_p^j$ | $T_I^j$ |
|---|---|---|
| $j = 1$ | -20 | 1.57 |
| $j = 2$ | -1 | 1.7 |
| $j = 3$ | -20.2 | 1.55 |

The controller algorithm was applied in the form (2.93), *i.e.*,

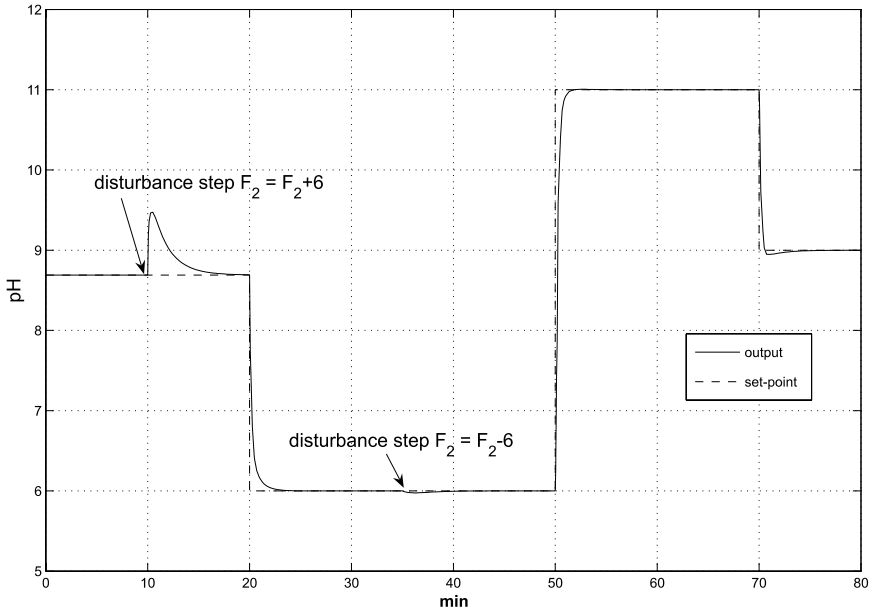


**Fig. 2.34.** Output trajectory in the $pH$ control system with the TS fuzzy PI controller

$$u(t) = \sum_{j=1}^{3} \tilde{w}^j(t) k_P^j \cdot e(t) + \sum_{j=1}^{3} \tilde{w}^j(t) \frac{k_P^j}{T_I^j} \cdot \int_0^t e(\tau) d\tau + \sum_{j=1}^{3} \tilde{w}^j(t) k_P^j T_D^j \cdot \frac{de(t)}{dt}$$

where

$$e(t) = pH(t)^{sp} - pH(t)$$

and $\tilde{w}^j(t)$ are normalized activation levels of the rules with antecedents in the form : "$pH(t)$ is $pH_j$", where $pH_j$, $j = 1, 2, 3$ are linguistic values "small", "medium" and "high" of the linguistic variable "pH value", corresponding to three fuzzy sets described by the membership functions shown in Fig. 2.33.

A chosen result of the control system simulation, corresponding to step changes of the set-point and of the disturbance (inflow rate $F_2$) is presented in Fig. 2.34. This is the result of a simulation with a continuous process and a discretized controller, with the sampling period $T_p = 0.02\,\text{min}$. □

## 2.4 Feedforward Compensation, Automatic Tuning

### Measured Disturbance and Feedforward Compensation

A designer of control systems should always obey the well-known general principle that *the influence of measured disturbances*, *i.e.*, uncontrolled but measured inputs, *should be compensated in a feedforward structure* – if only influence of disturbances on the controlled variables is significant and dynamics of the process enables effective realization of the feedforward path (roughly, influence of the manipulated input on the controlled output should not be slower then influence of the disturbance). Feedforward control is a control in an open-loop structure, therefore it is faster and more effective than in the closed-loop structure provided its mentioned applicability conditions are satisfied. Then, for control in the feedback loop, there remains reduction of influences of unmeasured disturbances and errors caused by inaccuracies in the process models used for the design of the feedback controller and feedforward compensator. The feedback control structure with measured disturbance compensation in the feedforward path is presented in Fig. 2.35.

The feedforward compensation of disturbances operates independently of the closed-loop feedback control, it does not influence the closed-loop stability conditions. The control signal $u$ consists of a feedback controller signal $u_c$ and feedforward compensator signal $u_f$. The design of a compensator algorithm is independent of the design of a feedback controller algorithm. Nonlinear TS fuzzy controllers considered in this chapter, both in their discrete and continuous versions, generate the control signal $u_c$. Control structures with these controllers can be easily supplemented by modules realizing compensation of the measured disturbances, as shown in Fig. 2.35. The design of a compensator algorithm is based on generation of a signal $u_f$ compensating the influence of the disturbance $z$ on the process output $y$. If $G$ and $G_z$ denote transfer functions of the control path and of the disturbance path, respectively, *i.e.*,

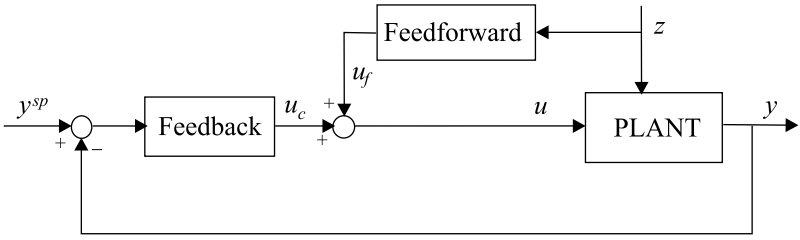


**Fig. 2.35.** Feedback-feedforward control structure with measured disturbance compensation in the feedforward path

$$y = Gu + G_z z$$

then the output signal of an ideal compensator should be equal to

$$u_f = -G^{-1} G_z z \tag{2.98}$$

Certainly, a difficulty in a the design of a compensator results from the fact that the inverse of the transfer function $G$ should be stable and the transfer function of the compensator must be physically implementable. Most important condition for a successful compensation is that time delay in the control path should not exceed the one in the disturbance path and that the transfer function of the control path must not have unstable zeros, see *e.g.*, [45, 3, 52]. Moreover, accuracy of the process model used for compensator design is important for quality of compensation – but this aspect is not critical for the structure from Fig. 2.35, as inaccuracies of compensation can be reduced or eliminated in the feedback loop.

To design a compensator it is necessary to have a process model. In this chapter nonlinear processes described by TS fuzzy models were considered. In the presence of measured disturbances significantly influencing the process behavior, the TS fuzzy models will also depend on these variables. The disturbances will therefore be present in the formulations of rule antecedents and consequents. Let us consider a case of discrete-time models described by difference equations. Denoting the value of a disturbance measured at $k$-th sampling instant by $z(k)$, the rules of a *TS fuzzy model used for the design of a fuzzy controller with disturbance compensation* can be written in the following very general form (compare with (2.36)):

$$\begin{aligned} \mathrm{R}_p^i: \quad \text{IF} \quad & y(k) \text{ is } A_0^i \text{ and } y(k-1) \text{ is } A_1^i \text{ and } \cdots \text{ and } y(k-n_R) \text{ is } A_{n_R}^i \\ & \text{and } z(k) \text{ is } Z_0^i \text{ and } z(k-1) \text{ is } Z_1^i \text{ and } \cdots \text{ and } z(k-p_R) \text{ is } Z_{p_R}^i \\ & \text{and } u(k-1) \text{ is } B_1^i \text{ and } \cdots \text{ and } u(k-m_R) \text{ is } B_{m_R}^i \end{aligned}$$

$$\begin{aligned}\text{THEN}\quad y^i(k+1) = {} & a_1^i y(k) + a_2^i y(k-1) + \cdots + a_{n_A}^i y(k-n_A+1) + \\ & + b_0^i u(k) + b_1^i u(k-1) + \cdots + b_{m_B}^i u(k-m_B) + \\ & + f_0^i z(k) + f_1^i z(k-1) + \cdots + f_{p_F}^i z(k-p_F) \end{aligned} \tag{2.99}$$

where $i = 1, \ldots, r$ indexes the rules (and corresponding fuzzy sets) in the domain of a TS model, $y(k)$, $z(k)$ and $u(k)$ are values of output, disturbance and control input at $k$-th sampling instant, $A_j^i \in \mathbb{Y}_j$, $B_j^i \in \mathbb{U}_j$, $Z_j^i \in \mathbb{Z}_j$, and $a_j^i$, $b_j^i$ and $f_j^i$ are coefficients of functions in the rule consequents. Elements of each set $\mathbb{Y}_j = \{Y_{j1}, ..., Y_{jr_{yj}}\}$ are fuzzy sets covering the domain of the variable $y(k-j)$, $j = 0, \ldots, n_R$, analogously $\mathbb{U}_j = \{U_{j1}, ..., U_{jr_{uj}}\}$ for $u(k-j)$, $j = 1, \ldots, m_R$ and $\mathbb{Z}_j = \{Z_{j1}, ..., Z_{jr_{zj}}\}$ (see comments after (2.36)). The form of the rule antecedent (2.99) looks rather complicated due to the assumed generality. However, in many applications there are only a few conditions there, in particular delayed variables are often not present, if there are process outputs $y$ then there are no process inputs $u$, *etc.*

The output of a fuzzy model is calculated according to the general formula

$$y(k+1) = \frac{\sum_{i=1}^{r} [w^i(k) y^i(k+1)]}{\sum_{l=1}^{r} w^l(k)} \tag{2.100}$$

wherte $w^i(k)$ are activation levels of individual rules (2.99) at sampling instant $k$,

$$w^i(k) = \prod_{j=0}^{n_R} \mu_{A_j^i}(y(k-j)) \prod_{j=0}^{p_R} \mu_{Z_j^i}(z(k-j)) \prod_{j=1}^{m_R} \mu_{B_j^i}(u(k-j)) \tag{2.101}$$

For each linear model in the rule consequents (2.99) both a feedback controller algorithm as well as a feedforward compensator algorithm are designed. Design of a controller was considered in Section 2.2.2 – the only difference caused by the presence of a measured disturbance can be an enlarged number of rules, due to a nonlinear dependence on an additional disturbance variable. Every local compensator algorithm is designed according to the general law of feedforward compensation, shortly recalled in (2.98). For the considered fuzzy modeling method it will be in the form of a difference equation. This way, we obtain the following rules of a *discrete, nonlinear TS fuzzy compensator*

$$\begin{aligned}\text{R}_f^i:\quad \text{IF}\quad & y(k) \text{ is } A_0^i \text{ and } y(k-1) \text{ is } A_1^i \text{ and } \cdots \text{ and } y(k-n_R) \text{ is } A_{n_R}^i \\ & \text{and } z(k) \text{ is } Z_0^i \text{ and } z(k-1) \text{ is } Z_1^i \text{ and } \cdots \text{ and } z(k-p_R) \text{ is } Z_{p_R}^i \\ & \text{and } u(k-1) \text{ is } B_1^i \text{ and } \cdots \text{ and } u(k-m_R) \text{ is } B_{m_R}^i \end{aligned}$$

$$\begin{aligned}\text{THEN}\quad u_f^i(k) = {} & g_1^i u_f(k-1) + g_2^i u_f(k-2) + \cdots + g_{m_G}^i u_f(k-m_G) + \\ & + h_0^i z(k) + h_1^i z(k-1) + \cdots + h_{p_H}^i z(k-p_H) \end{aligned} \tag{2.102}$$

where $i = 1, \ldots, r$ indexes rules, while $g_j^i$ and $h_j^i$ are coefficients of functions of the rule consequents describing local compensators. The nonlinear compensator output signal assumes a standard form for TS fuzzy structures

$$u_f(k) = \sum_{i=1}^{r} \tilde{w}^i(k) u_f^i(k) \tag{2.103}$$

where $\tilde{w}^i(k)$ are normalized activation levels of the rules (2.102). Compensation of the disturbance by each of the r local compensators is linear, but due to the fuzzy reasoning the entire TS fuzzy compensator defined by (2.102) and (2.103) is of course *nonlinear*.

**Automatic Tuning of a TS-FPID Controller**

In modern industrial PID controllers it is now the standard to have a software for automatic tuning (auto-tuning, self-tuning) installed, operating on the basis of a certain, previously programmed active experiment, usually a relay control phase, a pulse response or a multi-frequency response. By activating this software option the user automatically obtains settings calculated by the controller at the current operating point of the process, the user does not have to know the algorithm or assumed process model used for doing that.

If one applies the mentioned self-tuning of linear controllers, then in the first stage of the design of a TS fuzzy controller it will not be necessary to have local process models needed for a design of the rule consequents of a fuzzy controller. It will, however, be necessary to perform fuzzy partitioning and allocate positions of operating points in appropriately chosen centers of the obtained fuzzy sets (sub-domains). Then, self-tuning will be performed at each of these points.

In a case of a confidence in the settings of the PID controllers obtained by the automatic self-tuning procedure, sufficiently dense fuzzy partitioning (in relation to the process nonlinearity) and confidence in the reliability of the controller software implementing fuzzy reasoning, it is possible to test the TS-FPID controller straight on a real application. Such a solution was implemented in certain fuzzy controllers, *e.g.*, it was the case in EFTRONIK XF controller manufactured by PNEFAL company [114]. The user of this controller defines the number and position of operating points (up to ten), at each of them a self-tuning is activated by the user and on the basis of obtained parameters of local PID controllers a TS-FPID controller is created automatically, in its final, discrete form. Certainly, an important design parameter is also the controller sampling period.

# 3

# Model-based Predictive Control

Model-based predictive control (MPC) is the only one among the so-called *advanced control techniques* (usually understood as techniques more advanced than a standard PID control) which was tremendously successful in practical applications in recent decades, exerting a great influence on directions of development of industrial control systems as well as research in this area, see *e.g.*, [98, 1, 36, 94, 82, 115, 123, 11]. There are several reasons for this success. Firstly, the MPC algorithms can directly take into account constraints on both process inputs and outputs which often decide on the quality, effectiveness and safety of production. Secondly, they generate process inputs taking into account internal interactions within the process, due to the direct use of a model. Thus, they can be applied to processes with difficult dynamics and to multivariable control, even when numbers of manipulated and controlled variables are uneven. Thirdly, the principle of operation of these algorithms is comprehensible and relatively easy to explain to engineering and operator staff, which is a very important aspect when introducing new techniques into industrial practice.

## 3.1 The Principle of Predictive Control

The *general principle of predictive control* can be described as follows:
At each consecutive sampling instant $k$ (*i.e.*, at a continuous time $kT_p$, where $T_p$ denotes the controller sampling period), $k = 0, 1, ...$, having:

- a dynamic process model together with an assumed model of disturbances (uncontrolled process inputs) and models of constraints,
- measurements of current and past process outputs, together with past values of control inputs (manipulated variables),
- known or assumed trajectories of set-points for the controlled variables (controlled process outputs) for an assumed horizon of prediction,

the control inputs $u(k) = u(k|k), u(k+1|k), \ldots, u(k+N_u-1|k)$ are calculated, assuming $u(k+p|k) = u(k+N_u-1|k)$ for $p \geq N_u$, where $N_u$ is the *control*

*horizon*. The applied notation "$u(k+p|k)$" means a prediction of the control inputs for the future sampling instant $k+p$, performed at the current instant $k$. The control inputs are calculated in such a way as to minimize differences between the predicted controlled outputs $y(k+p|k)$ and the required reference values, the future set-points $y^{sp}(k+p|k)$, over the *prediction horizon* $N$ $(p = 1, 2, ..., N)$. Minimization of differences is understood in the sense of minimizing a selected criterion of control quality. Then, only the first element of the calculated sequence of control inputs is applied to the process, *i.e.*, the control input $u(k) = u(k|k)$. At the next sampling instant $(k+1)$ there occurs a new measurement of the process outputs and the whole procedure is repeated, with the prediction horizon of the same length $N$, but shifted by one step forward. Thus, the principle of a *receding horizon* is used, called also the repetitive control principle, see *e.g.*, [37]).

The principle of the predictive control, for the case of a SISO (single-input single-output) process, is presented in Fig. 3.1, where the horizontal axis represents the discrete time with $k$ being a current sampling instant at which a decision about the current process input signal $u(k) = u(k|k)$ is to be taken (with value constant over the whole sampling time interval $[kT_p, (k+1)T_p)$ ). The variables which are needed for calculation of the input value $u(k)$ are presented as appropriate trajectories of the process control input and controlled output. The figure presents two controlled output and two control input trajectories and a trajectory of the set-point, in particular:

- A predicted controlled output trajectory $y^0(k+p|k),\ p = 1, 2, ..., N$, corresponding to the situation where the process input is kept constant over the entire prediction horizon, with the value $u(k-1)$ calculated at the preceding sampling instant, *i.e.*, $u(k+p-1) = u(k-1)$ for each $p = 1, 2, ..., N$. Trajectories corresponding to this case, of both the output and the input, have been presented as dashed lines. The trajectory $y^0(k+p|k)$ defined in this way presents the future outputs as dependent on the *previous* inputs only, we have no influence on this trajectory at a current sampling instant $k$. Therefore, it is often described as a *free component of the predicted output trajectory* (in short: a free output trajectory).
- A predicted controlled output trajectory $y(k+p|k)$ dependent both on *past* and *future* control inputs, *i.e.*, on past inputs up to the last one $u(k-1)$ and future inputs $u(k+p-1|k)$, $p = 1, 2, ..., N-1$ which are calculated at a current sampling instant $k$. It is assumed that the control horizon $N_u$ (precisely: horizon of process control input variability) can be shorter than the prediction horizon $N$, $N_u \leq N$. The trajectories listed are presented as continuous curves in Fig. 3.1. Thin fragments of the curves denote predicted parts of the input and output trajectories, whereas thick parts denote the output trajectory which has already been realized (measured) and the input trajectory applied earlier together with the last element being applied to the plant at the current sampling instant.

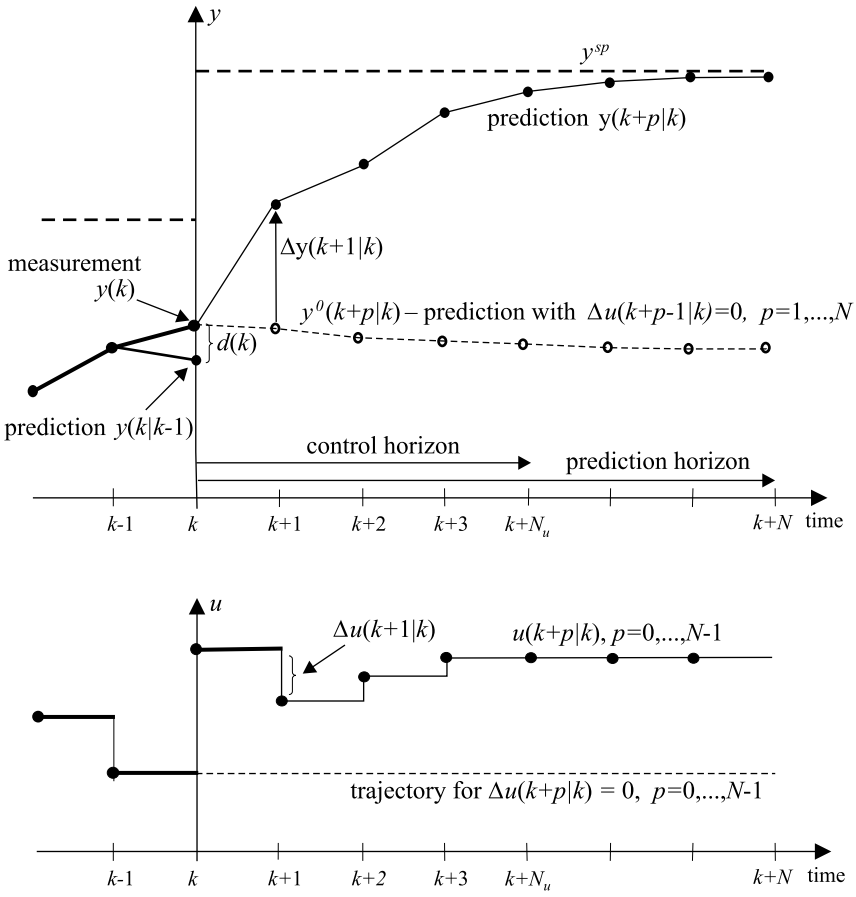


**Fig. 3.1.** Illustration of the principle of predictive control

- A known or foreseen trajectory of the set-points for the controlled output $y^{sp} = y^{sp}(k+p|k)$, $p = 1, 2, ..., N$, presented as a thicker dashed line. The set-point for the process output presented in Fig. 3.1 underwent a step change at sampling instant $k$ and then remains constant. Generally, it can be varying over the prediction horizon.

A model of the process used for calculation of the future process control inputs is usually only an approximation of reality. Further, there is uncertainty in the uncontrolled inputs, which can be inaccurately measured or not measured at all. Therefore, the output predictions usually differ from the (later) measured values. This fact is depicted in Fig. 3.1 as an unmeasured disturbance $d(k)$, $d(k) = y(k) - y(k|k-1)$, occurring at the process output at sampling instant $k$,

where $y(k|k-1)$ is the process output value predicted for the sampling instant $k$ at the preceding one, $(k-1)$. The dependence of the trajectory of the process output (evaluated at sampling instant $k$) on the currently measured value $y(k)$, and not on the value $y(k|k-1)$, means that a *discrete output-feedback* is applied in the control system.

Determination of the control inputs for the present sampling instant $k$ and for the future instants $k+p$ in the control horizon, $p=1,...,N$, is realized in the predictive algorithms on the basis of a process model, by minimizing a selected cost function describing the control quality over the prediction horizon. A prime component of this function is the *cost of deviations of the predicted outputs from the set-points*, *i.e.*, the cost of predicted control errors. Moreover, it is also typical to include into the cost function penalties for control input changes. Considering the two mentioned components, the following most commonly used *quadratic cost function* (*objective function*) of the predictive control can be formulated and used for calculation of the optimal process input trajectory over the control horizon:

$$\begin{aligned} J(k) &= \sum_{p=N_1}^{N} (y^{sp}(k+p|k) - y(k+p|k))^T \mathbf{\Psi}(p)(y^{sp}(k+p|k) - y(k+p|k)) \\ &\qquad + \sum_{p=0}^{N_u-1} \triangle u(k+p|k)^T \mathbf{\Lambda}(p) \triangle u(k+p|k) \\ &= \sum_{p=N_1}^{N} \|y^{sp}(k+p|k) - y(k+p|k)\|^2_{\mathbf{\Psi}(p)} + \sum_{p=0}^{N_u-1} \|\triangle u(k+p|k)\|^2_{\mathbf{\Lambda}(p)} \end{aligned} \tag{3.1}$$

where vectors $y^{sp}(k+p|k)$ and $y(k+p|k)$ are of a dimensionality $n_y = \dim y$ (the number of the controlled outputs), while the vector of input increments $\triangle u(k+p|k)$ is of a dimensionality $n_u = \dim u$.

In (3.1) the differences $y^{sp}(k+p|k) - y(k+p|k)$ between the set-points and the predicted outputs are considered, starting from $k+N_1$ until the end of the prediction horizon $N$, where $1 \leq N_1 \leq N$. A value $N_1 > 1$ is reasonable if there is a delay in the process causing a lack of reaction of the outputs at first $N_1 - 1$ sampling instants, $k+1,...,k+N_1-1$, to the change of the control input at instant $k$. Of course, assuming $N_1 = 1$ is not an error also in the situation with a delay – only the first $N_1 - 1$ components of the first sum in the cost function (3.1) will then be unnecessarily calculated during the optimization process, not being dependent on the calculated inputs. However, sometimes in the literature $N_1 = 1$ is assumed for uniformity of the notation, which does not lead to loss of correctness, as it has just been mentioned. The length of the control horizon $N_u$ must satisfy the constraints $0 < N_u \leq N$. It is usually assumed that $N_u < N$, as this results in a decreased dimensionality of the controller optimization problem, and thus leads to smaller computational load.

The matrix $\mathbf{\Psi}(p)$ is a matrix of weights enabling scaling of the influence of different components of the error vector $y^{sp}(k+p|k) - y(k+p|k)$ in the cost function, it is a diagonal matrix. If matrices $\mathbf{\Psi}(p)$ are different for different values of $p$ (*i.e.*, for different future sampling instants), then there is also a scaling over the prediction horizon. If scaling of the control errors is not needed at all, then the simplest case is obtained, $\mathbf{\Psi}(p) = \mathbf{I}$, where $\mathbf{I}$ is a unity matrix of a dimension $n_y \times n_y$. The role of the matrix $\mathbf{\Lambda}(p)$ is, in turn, not only to introduce a scaling of individual components of the vector of control input moves, but first of all to introduce a scaling of the whole second sum in (3.1) against the first one representing the predicted control errors. In the simplest case without scaling over the prediction horizon as well as between individual components of the vector of the input moves, we obtain $\mathbf{\Lambda}(p) = \lambda\mathbf{I}$, where $\mathbf{I}$ is a unity matrix of a dimension $n_u \times n_u$. In a case when $\mathbf{\Psi}(p) = \mathbf{I}$ and $\mathbf{\Lambda}(p) = \lambda\mathbf{I}$, the cost function (3.1) takes the following simpler and commonly met form (see *e.g.*, [123]):

$$J(k) = \sum_{p=N_1}^{N} \|y^{sp}(k+p|k) - y(k+p|k)\|^2 + \lambda \sum_{p=0}^{N_u-1} \|\triangle u(k+p|k)\|^2 \qquad (3.2)$$

where the scalar $\lambda \geq 0$ defines in fact a ratio of the weight attributed to damping of the input moves versus the (unity) weight attributed to a reduction of the control errors. Let us emphasize that assuming $\lambda = 0$ is possible, but when there are no constraints on values and rate of changes of the process inputs this often leads to practically unacceptable controller properties, particularly to huge input changes and insufficient robustness against modeling errors.

A different, additional mechanism of damping the input increments and enforcing a desired, more calm trajectory of the outputs is to apply a *reference trajectory* $y^{ref}(k+p|k)$ in the cost function (3.1), in place of the trajectory of the set-points $y^{sp}(k+p|k)$. The reference trajectory constitutes a certain intermediate trajectory between the currently measured value of the controlled outputs $y(k)$ and a trajectory of the set-points. It will be discussed in more detail in Section 3.6.2.

Among the values present in the cost function (3.1), or its simplified and often sufficient form (3.2), there are (see also Fig. 3.1):

- known or directly forecasted vectors of the set-points for the controlled variables $y^{sp}(k+p|k)$,
- process inputs increments $\triangle u(k+p|k)$ which constitute decision variables of the controller optimization task,
- predicted values of the controlled variables $y(k+p|k)$ dependent on previous outputs and inputs and on actually calculated future input increments $\triangle u(k+p|k)$.

To calculate the values $y(k+p|k)$ in the prediction horizon, $p = N_1, ..., N$, it is necessary to have a *process model*. Generally, this can be a nonlinear model. So far, the MPC algorithms with *linear process models* have had the biggest

importance. First of all, a range of direct applications of these algorithms is fairly wide. Secondly, they constitute a basis of a construction of relatively simple and efficient nonlinear algorithms with linearized models, as it will be presented in further parts of this chapter.

In a linear case, applying the principle of superposition, it is possible to present the trajectory of predicted outputs $y(k+p|k)$ in the form of a sum of a *free trajectory* $y^0(k+p|k)$ dependent only on the realized (past) process inputs and a trajectory $\triangle y(k+p|k)$ dependent only on the decision variables (current and future inputs $u(k+p|k)$). Thus, the trajectory $\triangle y(k+p|k)$, $p = N_1, ..., N$ is called a *forced output trajectory* (precisely, it is a forced component of the predicted output trajectory). Thus we have

$$y(k+p|k) = y^0(k+p|k) + \triangle y(k+p|k), \quad p = N_1, ..., N. \tag{3.3}$$

The above partition is convenient, though not necessary for a realization of a predictive control algorithm, because the values $y^0(k+p|k)$, as dependent only on the past of the process, are calculated by the control algorithm at a current sampling instant $k$ *only once*, they remain then as *fixed parameters* in the further optimization of the input changes. On the other hand, the increments $\triangle y(k+p|k)$, as dependent on the current and future input changes $\triangle u(k+p|k)$, are calculated *many times* in the process of the numerical optimization. In fact, a model of a dependence of these increments on input changes is used, having the functional form of the dependence of $\triangle y(k+p|k)$ on $\triangle u(k+j|k)$, $p = N_1, ..., N$, $j = 0, 1, ..., \min\{p, N_u\} - 1$. Considering the presented decomposition of the predicted trajectory of the outputs, the cost function (3.1) can be rewritten in the following form:

$$J(k) = \sum_{p=N_1}^{N} \left\| [y^{sp}(k+p|k) - y^0(k+p|k)] - \triangle y(k+p|k) \right\|^2_{\mathbf{\Psi}(p)} + \\ + \sum_{p=0}^{N_u-1} \| \triangle u(k+p|k) \|^2_{\mathbf{\Lambda}(p)} \tag{3.4}$$

and the cost function (3.2) in the form

$$J(k) = \sum_{p=N_1}^{N} \left\| [y^{sp}(k+p|k) - y^0(k+p|k)] - \triangle y(k+p|k) \right\|^2 + \\ + \lambda \sum_{p=0}^{N_u-1} \| \triangle u(k+p|k) \|^2 \tag{3.5}$$

Let us denote linear models, *i.e.*, matrices (since a linear mapping between two finite dimensional vector spaces can always be presented as a matrix), used for the prediction of the elements $\triangle y(k+p|k)$ of the forced output trajectory, by $\mathbf{M}_p$,

$$\triangle y(k+p|k) = \mathbf{M}_p[\triangle u(k|k)^T \ \triangle u(k+1|k)^T \ \cdots \ \triangle u(k+p-1|k)^T]^T, \quad p = 1, ..., N_u$$

$$\triangle y(k+p|k) = \mathbf{M}_p[\triangle u(k|k)^T \ \triangle u(k+1|k)^T \ \cdots \ \triangle u(k+N_u-1|k)^T]^T, \quad p = N_u+1, ..., N \quad (3.6)$$

Due to linearity of the models (3.6), the minimization of (3.4), or of (3.5), is a minimization of a *convex quadratic function*, strictly convex if $\mathbf{\Lambda}(p) > 0$ and $\mathbf{\Psi}(p) \geq 0$ (*e.g.*, if $\lambda > 0$ when $\mathbf{\Lambda}(p)=\lambda\mathbf{I}$ and $\mathbf{\Psi}(p)=\mathbf{I}$). Therefore, this is a problem which has a unique, global minimum. Moreover, this problem can easily be solved analytically if there are no additional inequality constraints – or can be relatively fast and reliably (most important for on-line applications) computed numerically when such constraints occur.

If, however, a *nonlinear process model* is used for the prediction of the controlled outputs $y(k+p|k)$, $p = N_1, ..., N$, then the situation is more difficult. The free trajectory $y^0(k+p|k)$ can also be computed relatively easy, by assuming zero increments of the current and future inputs to the nonlinear process model. However, in general it is not possible to decompose a nonlinear model into independent components generating a free output trajectory and a forced output trajectory. Therefore, the nonlinear dependence of the predicted output trajectory $y(k+p|k)$ on the decision variables $\triangle u(k+p|k)$, $p = 0, 1, ..., N_u - 1$, is a crucial problem. Nonlinearity of this dependence implies that the minimization of the controller cost function becomes a *nonlinear, non-convex optimization problem*, in general. For such problems it is generally not possible to find an analytical solution even in cases without additional inequality constraints. Moreover, the process of numerical optimization is more difficult and much less effective – it is common to encounter local minima, it is difficult to estimate the time needed for the optimization and the result, in the form of a minimizing point found, is not guaranteed. The questions of predictive control algorithms with nonlinear process models used will be discussed in more detail in Section 3.5.

In the first years of the development of the MPC algorithms linear process models were used, only later the nonlinear ones – but the latter primarily for prediction of free trajectories of the outputs, $y^0(k+p|k)$. Commercial MPC programs did use linear models and still do mainly use such models for modeling the dependencies of $\triangle y(k+p|k)$ as functions of $\triangle u(k+p|k)$.

An optimal sequence of process input increments, $\triangle u(k|k)$, $\triangle u(k+1|k)$, $\triangle u(k+2|k)$, ..., $\triangle u(k+N_u-1|k)$, is obtained in predictive control algorithms by *minimization of the cost function* (3.1). During the initial stage of development of the MPC algorithms it was usually optimization without additional explicit constraints, but also an additional mechanism of influencing the shape of a solution trajectory was used, by applying a reference trajectory (see Section 3.6.2). For a linear process model (3.6), the optimization without constraints leads to an analytical solution, to a formula possible for implementation and calculation in real time using the control equipment of even fairly

small computing power. Development of the microprocessor technology and the following increase of calculation possibilities enabled real time numerical solutions of linear-quadratic optimization problems with additional inequality constraints. This became one of the main reasons for a development and a significant increase of the number of industrial applications of the predictive control algorithms.

The following *constraints* are important in applications and are possible to be considered directly in the MPC algorithms:

- Constraints on *values (amplitudes) of process control inputs*:

$$u_{\min} \leq u(k+p|k) \leq u_{\max}, \quad p = 0, 1, ..., N_u - 1 \tag{3.7}$$

- Constraints on *increments of process control inputs*:

$$-\triangle u_{\max} \leq \triangle u(k+p|k) \leq \triangle u_{\max}, \quad p = 0, 1, ..., N_u - 1 \tag{3.8}$$

- Constraints on *values of controlled outputs*, which in a concise form can be written as

$$y_{\min} \leq y(k+p|k) \leq y_{\max}, \quad p = N_1, N_1 + 1, ..., N \tag{3.9}$$

Constraints on controlled outputs are sometimes imposed not for the entire prediction window $[N_1, N]$, but only for a narrower *constraint window* $[N_{cw1}, N_{cw}]$, where$N_1 \leq N_{cw1} < N_{cw} \leq N$. This is one of methods to prevent the admissible set of the optimization problem from becoming empty, which can happen when it is not possible to satisfy the constraints on a certain output at first steps of the prediction horizon, *e.g.*, if the currently measured value of the output does significantly exceed the constraints due to a violent change in a disturbance value. We shall return to this question in Section 3.6.2, where the use of the concept of a constraint window will be explored.

The form (3.9) presents the most general case of *two-sided constraints*, known also as *band constraints* or *range constraints*. Of course, *one-sided constraints* constitute a special case, such as *e.g.*,

$$y(k+p|k) \leq y_{\max}, \quad p = N_1, N_1 + 1, ..., N$$

The constraining values can also be time-dependent, a situation encountered in the case of output constraints. The dependence on time can be either in the sense of a position in the prediction horizon, defined as dependence on $p$, *e.g.*, $y_{\max} = y_{\max}(p)$, $p = N_1, N_1 + 1, ..., N$, or as dependence on the current time $k$ (this last case leads to a *non-stationary* predictive control algorithm).

In applications there are also *constrained, but uncontrolled output variables*, which are also called *constraint variables*, see *e.g.*, [11]. Denoting these variables by $y^{ct}(k)$, the constraints on them can be formulated in a general form analogous to (3.9):

$$y^{ct}_{\min} \leq y^{ct}(k+p|k) \leq y^{ct}_{\max}, \quad p = N_1, N_1 + 1, ..., N \tag{3.10}$$

The uncontrolled outputs $y^{ct}(k)$ with constrained values must be directly or indirectly measured, since their predicted values $y^{ct}(k+p|k)$ must be calculated using appropriate models, in the same way as the values $y(k+p|k)$ are calculated. In this chapter controlled and uncontrolled output variables will not be distinguished until necessary, in order to avoid an unnecessary complication of the resulting notation.

We have thus defined all elements of the *optimization problem* which is solved at any step $k$ of a general MPC algorithm (with constraints), in order to evaluate an optimal trajectory of the process inputs over the control horizon $N_u$. For a linear process model this optimization problem can be formulated in the following way:

$$\min_{\triangle u(k|k),\ldots,\triangle u(k+N_u-1|k)} \{ \sum_{p=N_1}^{N} \|[y^{sp}(k+p|k) - y^0(k+p|k)] + \\ -\triangle y(k+p|k)\|^2_{\mathbf{\Psi}(p)} + \sum_{p=0}^{N_u-1} \|\triangle u(k+p|k)\|^2_{\mathbf{\Lambda}(p)} \}$$

$$\text{subj. to: the constraints (3.6), (3.7), (3.8), (3.9)} \tag{3.11}$$

An important problem when solving the optimization problem (3.11), a problem with inequality constraints on values of control inputs and controlled outputs, is the possibility that at certain sampling instants $k$ the *set of its feasible solutions (feasible set) can become empty*. This means that it would then be impossible to satisfy, simultaneously, the constraints (3.7), (3.8), (3.9) and the model equations (3.6), for certain current and previous values of the process outputs, inputs and disturbances. Let us notice that such a situation can occur only when there are constraints on the process outputs. Constraints on values and rates of change of the process inputs themselves cannot cause the feasible set to become empty. Certainly, only if these constraints are not initially formulated in a way leading to a contradiction, which is an erroneous formulation requiring a correction. Situations with a lack of admissible solutions should be avoided in practical implementations of the MPC algorithms, perhaps except for specially treated isolated cases. We shall return to the question of feasibility of the controller optimization problem (*i.e.*, non-emptiness of the set defined by its constraints) in Section 3.6.2.

From a historical perspective, the first practical applications of predictive control algorithms with a receding horizon took place in the 1970s, developed independently at different sites – although the idea itself was in fact formulated earlier [76], see [94]. Dates of first publications are not very authoritative in this case, as predictive algorithms were proposed and implemented primarily in the industry, where successful applications of new ideas often precede their publication. The first algorithms, based on a common general rule of the predictive control, differed mainly in the way the process was modeled, also different were details of formulations of the cost function and of the optimization task, methods of solving it, and approaches to treating the constraints.

First publications concerning an industrial application of an MPC algorithm were written by Richalet *et al.* [119, 120], concerning the algorithm called *Model Predictive Heuristic Control* (MPHC), implemented in a commercial software under the name IDCOM (*IDentification and COMmand*), more commonly known as a MAC algorithm (*Model Algorithmic Control*). For prediction of the controlled outputs, a linear process model was used in the form of a finite impulse response; the cost function was formulated as a sum of squared deviations of the predicted trajectory from a reference trajectory. The latter is an "intermediate" trajectory between the free response trajectory of a model and a set-point trajectory, it begins at the currently measured values of the outputs and converges to the trajectory of the set-points over the prediction horizon, its dynamics is usually of first order, see Section 3.6.2. The constraints were taken into account in the MAC algorithm, the process inputs were calculated by a heuristic iterative algorithm. Large emphasis was placed on a simplicity of tuning of the algorithm.

The *Dynamic Matrix Control* (DMC) algorithm, developed in the early 1970s by staff at Shell Oil and with an initial application in 1973, was also one of the first, being published later along with information about successful applications [29, 113]. This algorithm became very popular in commercial applications of predictive control algorithms. The DMC algorithm uses a linear process model in the form of a step response and a quadratic cost function with penalty components for changes of the process control inputs (as in (3.5)).

It is considered that the IDCOM and DMC algorithms represent the *first generation of the MPC algorithms*, see *e.g.*, [1, 52, 82]. These algorithms exerted an enormous influence on the development of practice and theory of the industrial control. In these algorithms the control inputs were not yet computed on-line by a numerical solving of a constrained optimization problem at each sampling instant.

The main drawback of the MPC algorithms of the first generation was treating the constraints in an approximate, heuristic way. In the DMC algorithm this disadvantage was removed proposing its version under the name *Quadratic Dynamic Matrix Control* (QDMC) [48]. At every step of the algorithm the process control inputs are calculated as a numerical solution of a minimization problem of a quadratic cost function (the same as in DMC) subject to linear constraints on process inputs and outputs, *i.e.*, the quadratic programming problem. The DMC algorithm in a QDMC version is considered to be a representative of the *second generation of the MPC algorithms*. A characteristic feature of these algorithms is an on-line numerical solving of a quadratic programming problem, ensuring direct consideration of inequality constraints.

The *Generalized Predictive Control* (GPC) algorithm proposed much later by Clark *et al.* turned out to be very popular [27, 26], it used a process model in the form of discrete transfer functions (or, equivalently, difference equations). This method of modeling allowed the consideration of a wider class of disturbance models, in comparison to those used in the DMC and MPHC

algorithms. An interesting development of the GPC algorithm is the CRHPC (*Constrained Receding Horizon Predictive Control*) proposed by Clark and Scattolini, it is one of the first formulations of the predictive control algorithms with a finite, receding horizon and with a theoretically established stability of the feedback control loop [28].

The *Shell Multivariable Optimizing Controller* (SMOC) [86] was the first to employ a process model in the form of state equations – an approach which is currently dominant in research studies concerning the MPC algorithms. A state observer (Kalman filter) was used for the estimation of state variables and unmeasured disturbances. Moreover, a differentiation between controlled and measured process outputs was introduced.

The SMOC is already an example of a modern, *third generation* MPC algorithm, just like the algorithms IDCOM-M by Setpoint, RMPCT by Honeywell, 3dMPC by ABB and others. A short review of selected commercial implementations of the MPC algorithms can be found in [82], a comprehensive survey is given in an excellent paper [115].

In the development conducted over the last years and presently, in the research and in applications of the MPC algorithms, the problems that dominate are those of considering nonlinear process models, stability, robustness to uncertainty. It can be said that these topics distinguish the MPC algorithms of the *fourth generation*.

In the following sections of this chapter selected predictive control algorithms will be presented. The choice was done taking into account, first of all, practical importance. First we shall present in detail the construction and basic features of a predictive control algorithm with a linear process model, on an example of the famous DMC algorithm. This choice was dictated also by the fact that on the basis of a process model used in the DMC algorithm, in the form of a step response, it is easy and also most natural to derive and interpret dependencies describing the predictive control. The step response model is formulated in the time domain, which is natural for formulation of the MPC algorithms. The DMC algorithm was developed in the petrochemical industry, it was one of the first successfully applied in practice, and then became a very popular solution. Moreover, software packages employing the DMC philosophy are still commonly distributed in industrial applications, as identification of the process model in the form of its step responses is one of the simplest, and yet most efficient, methods. We shall also present the GPC algorithm in detail, it uses the process model in the form of linear difference equations (which are equivalent to discrete transfer functions). The MPC algorithm with the process model in the form of linear state equations will also be considered. However, it will be presented emphasizing common features and differences between the earlier introduced and more thoroughly discussed DMC and GPC algorithms.

The predictive control algorithms with a linear process model will be a starting point to the presentation of the MPC algorithms using nonlinear process models. In this area, after presenting a general approach, we shall

concentrate mainly on algorithms using on-line repetitive linearizations of a nonlinear model – as this approach leads to fast and reliable implementations, and at the same time is very efficient for a wide range of nonlinear processes. These algorithms will be discussed in more detail for the case of processes described by nonlinear Takagi-Sugeno fuzzy models, as it is then particularly easy to develop and implement nonlinear algorithms with linearizations.

The questions of *stability* of the predictive control algorithms and that of suitable modifications which guarantee stability will also shortly be presented. We shall touch on an important question of the parameter tuning, as well as the specific and important question of the design which ensures non-emptiness of the admissible solution set of the MPC controller optimization problem.

## 3.2 Dynamic Matrix Control (DMC) Algorithm

### 3.2.1 Output Predictions Using Step Response Models

#### Single-input Single-output (SISO) Processes

In the DMC algorithm the process dynamics is modeled by discrete step responses which describe reactions of the process outputs on unit step changes of process inputs. Using a response of a process output on an input step is in many applications a convenient way of modeling the dynamics. Figure 3.2 presents an example of a step response of a first order process with a delay, where the delay $\tau = 2T_p$ ($T_p$ – the sampling period). Both a real measured response is given, as well as a response after perfect filtering of a measurement noise (thick curve). Knowing the discrete step response of the process output in the form of a set of the coefficients $\{s_1, s_2, s_3, ...\}$ it is possible to model a discrete response of the output on any discrete control input signal

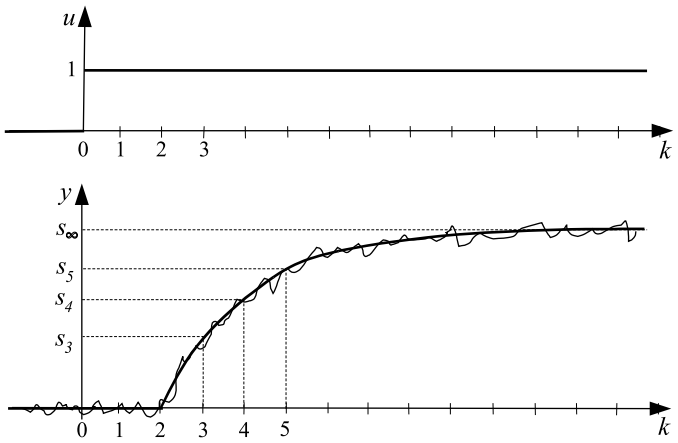


**Fig. 3.2.** Example of a response of the output $y$ on a step change in the input $u$

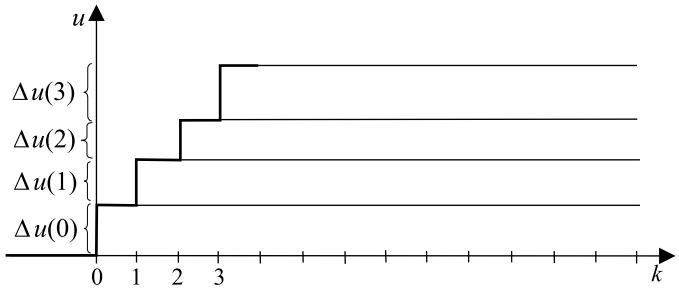


**Fig. 3.3.** A representation of a discrete signal as a sum of step signals with different amplitudes $\Delta u(k)$.

(with zero-order hold). It is because such a signal can be treated as a sum of steps beginning at consecutive sampling instants, with amplitudes equal to changes in the input signal – as it is presented in Fig. 3.3. Therefore, using the principle of superposition we can write:

$$\begin{aligned}
y(1) &= y(0) + s_1 \triangle u(0) \\
y(2) &= y(0) + s_2 \triangle u(0) + s_1 \triangle u(1) \\
y(3) &= y(0) + s_3 \triangle u(0) + s_2 \triangle u(1) + s_1 \triangle u(2) \\
y(4) &= y(0) + s_4 \triangle u(0) + s_3 \triangle u(1) + s_2 \triangle u(2) + s_1 \triangle u(3)
\end{aligned}$$

and so on, for the following steps. Therefore, for any given $k = 1, 2, 3, ...$ we obtain the following dependence

$$y(k) = y(0) + \sum_{j=1}^{k} s_j \triangle u(k-j) \tag{3.12}$$

Using (3.12), we can write for the sampling instant $k + p$

$$y(k+p) = y(0) + \sum_{j=1}^{k+p} s_j \triangle u(k+p-j) \tag{3.13}$$

The formula (3.13) will be used for derivation of the *prediction equations* in the DMC algorithm. As earlier, we will denote by $y(k+p|k)$ a value of the output predicted at the current sampling instant $k$ for a future instant $k+p$. Analogously, $\triangle u(k+p|k)$ denotes a value of the input change evaluated at sampling instant $k$ for a future instant $k+p$. The prediction $y(k+p|k)$ is equal to the output value evaluated from the model (3.13) supplemented by a disturbance value $d(k+p|k)$, forecasted for the same sampling instant $k+p$,

$$y(k+p|k) = y(0) + \sum_{j=1}^{p} s_j \triangle u(k+p-j|k) + \sum_{j=p+1}^{k+p} s_j \triangle u(k+p-j) + d(k+p|k)$$

The disturbance value at sampling instant $k$ is assumed to be equal to the difference of the measured output value $y(k)$ and the value calculated from the model

$$d(k) = y(k) - \left[ y(0) + \sum_{j=1}^{k} s_j \triangle u(k-j) \right] \tag{3.14}$$

In the DMC algorithm a lack of knowledge about future changes of the disturbances on a prediction horizon is assumed, therefore the following model is used (called the *constant output disturbance model*, or the *DMC disturbance type model*, see *e.g.*, [82])

$$d(k+1|k) = d(k+2|k) = \cdots = d(k+N|k) = d(k)$$

*i.e.*, in the prediction horizon the disturbance value is assumed to be constant and equal to the value (3.14) determined at sampling instant $k$. Observe that this disturbance prediction corresponds to the optimal prediction of the integrated white noise added to the process output, $d(k)$ can be treated as an integrated white noise signal (Wiener process).

Joining the equations obtained above we get the following formula for the output predictions

$$\begin{aligned} y(k+p|k) = y(k) &+ \sum_{j=1}^{p} s_j \triangle u(k+p-j|k) + \\ &+ \sum_{j=p+1}^{k+p} s_j \triangle u(k+p-j) - \sum_{j=1}^{k} s_j \triangle u(k-j), \quad p = 1, ..., N \end{aligned}$$

where $y(k)$ is the *output value measured at sampling instant* $k$, and $\triangle u(k-1)$, $\triangle u(k-2), ...$ are the control input changes determined and applied to the plant at preceding instants. Performing further manipulations we get

$$\begin{aligned} y(k+p|k) &= \sum_{j=1}^{p} s_j \triangle u(k+p-j|k) + \\ &+ y(k) + \sum_{j=1}^{k} s_{j+p} \triangle u(k-j) - \sum_{j=1}^{k} s_j \triangle u(k-j) \\ &= \sum_{j=1}^{p} s_j \triangle u(k+p-j|k) + y(k) + \sum_{j=1}^{k} (s_{j+p} - s_j) \triangle u(k-j) \end{aligned}$$

The first sum on the right hand side of the obtained equation depends on current and future control input changes, $\triangle u(k|k)$, $\triangle u(k+1|k), ...$, which are decision variables of the controller optimization problem. Therefore, we call this part of the predicted output trajectory a *forced component* of this trajectory:

$$\triangle y(k+p|k) = \sum_{j=1}^{p} s_j \triangle u(k+p-j|k), \quad p = 1, 2, ..., N \tag{3.15}$$

The remaining part of the predicted output trajectory is a *free component* dependent only on the previous changes of the control input and is given by the formula

$$y^0(k+p|k) = y(k) + \sum_{j=1}^{k} (s_{j+p} - s_j) \triangle u(k-j), \quad p = 1, 2, ..., N \tag{3.16}$$

For stable processes without integrated or runaway responses (*i.e.*, for asymptotically stable processes, with self-regulating outputs) the output stabilizes, after a step change in the input, at a certain value $s_\infty$, $\lim_{k\to\infty} s_k = s_\infty$, see Fig. 3.2. Therefore, it is enough to know a finite number, say $D$, of coefficients of the step response, *i.e.*, the number of steps after which the value of the step response can be treated as constant (and equal to the static process gain $k_m$ since the input was assumed to be a unit step starting from zero value). Thus, $D$ will be called a *horizon of the process dynamics*. The following estimation usually proves correct

$$D \cong (T_0 + (3 \div 4)T)/T_p$$

where $T_0$ is a process time delay and $T$ a process dominant time constant. If $s_j = k_m$ for $j \geq D$, then (3.16) takes the following form

$$y^0(k+p|k) = y(k) + \sum_{j=1}^{D-1} (s_{j+p} - s_j) \triangle u(k-j), \quad p = 1, 2, ..., N \tag{3.17}$$

because $s_{j+p} - s_j = 0$ for $j \geq D$. The presented method of process modeling in the form of finite step responses is applicable for asymptotically stable processes. However, in the industrial practice an important role is played by stable processes with integration. For this class of processes it is easy to generalize the considered method of modeling – by using an *incremental step response* instead of the standard step response considered above. Namely, dependence of increments of the process outputs $\triangle y(k) = y(k) - y(k-1)$, $k = 1, 2, 3, ...$ on increments of the process inputs will be modeled. Assuming a steady state for $k < 0$ and a unit step in the control input at sampling instant $k = 0$, $\triangle u(0) = 1$, we have

$$\begin{aligned}
\triangle y(1) &= y(1) - y(0) = \tilde{s}_1 = s_1 \\
\triangle y(2) &= y(2) - y(1) = \tilde{s}_2 = (s_2 - s_1) \\
\triangle y(3) &= y(3) - y(2) = \tilde{s}_3 = (s_3 - s_2) \\
&\vdots
\end{aligned}$$

where $\tilde{s}_k$ denote elements of the incremental step response, and $s_k$ elements of the previously introduced standard step response. For a stable plant with a single integration, differences between the subsequent output values in the standard step response become constant after the disappearance of a transient process corresponding to the dynamics horizon $D$ – then the final constant value of the incremental step response is achieved, a *speed gain* $\tilde{k}_m$. For any given discrete control input signal treated as a sum of steps we obtain the following formula (corresponding to (3.12) in the previous standard case)

$$\triangle y(k) = \sum_{j=1}^{k} \tilde{s}_j \triangle u(k-j) \tag{3.18}$$

All considerations which will further be presented, concerning the design of the MPC algorithm for an asymptotically stable process modeled by the step response, can easily be extended onto a process with single integration, modeled by an incremental step response.

### Multivariable (MIMO) Processes

Consider now a multivariable, *multi-input multi-output* (MIMO) process, with $n_y$ controlled outputs and $n_u$ control inputs. Let us consider first a process model in the form of a set of $n_y \cdot n_u$ finite step responses $\{s_l^{ij},\ l = 1, 2, ..., D\}$, where $i$ indexes controlled outputs, $i = 1, 2, ..., n_y$, and $j$ indexes process control inputs, $j = 1, 2, ..., n_u$. Therefore, components of the vector $s^{ij} = [s_1^{ij}\ s_2^{ij}\ ...\ s_D^{ij}]$ are elements of a step response of the $i$-th output to a unit step on the $j$-th input (when all other inputs are kept constant). The dynamics horizon $D$ is taken as common for all responses, *i.e.*, it can be assumed that $s_l^{ij} = \text{const.}$ for $l \geq D$.

The presented modeling of a multivariable process in the form of $n_y \cdot n_u$ vectors of finite step responses $[s_1^{ij}\ s_2^{ij}\ ...\ s_D^{ij}]$, $i = 1, 2, ..., n_y,\ j = 1, 2, ..., n_u$ is natural. In practice, it is a natural and basic method of identification of the step responses to perform a step change of one input (with all other inputs kept unchanged) and to register the subsequent output values – repeating this procedure, consequently, for all process inputs, $j = 1, 2, ..., n_u$. However, it is convenient to transform the presented multivariable model consisting of a set of vectors of step responses into a different, equivalent form, which will make the design of a DMC controller more clear and the resulting formulae much simpler. In order to do this we shall define the following matrices

$$\mathbf{S}_l = \begin{bmatrix} s_l^{11} & s_l^{12} & s_l^{13} & \cdots & s_l^{1n_u} \\ s_l^{21} & s_l^{22} & s_l^{23} & \cdots & s_l^{2n_u} \\ s_l^{31} & s_l^{32} & s_l^{33} & \cdots & s_l^{3n_u} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_l^{n_y 1} & s_l^{n_y 2} & s_l^{n_y 3} & \cdots & s_l^{n_y n_u} \end{bmatrix}, \quad l = 1, 2, ..., D \tag{3.19}$$

Each matrix $\mathbf{S}_l$ consists of coefficients $s_l^{ij}$ of all the step responses at the same sampling instant $l$, $i = 1, 2, ..., n_y$, $j = 1, 2, ..., n_u$. Therefore, the plant dynamics can be represented by $D$ matrices $\mathbf{S}_l$ of dimension $n_y \times n_u$, instead of $n_y \cdot n_u$ vectors of length $D$.

The set of matrices $\mathbf{S}_l$ can be treated as a *multivariable (matrix) step response* $\{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_D\}$. This model representation makes it possible *to directly apply all the formulae obtained so far for the SISO processes to the case of a MIMO process*, only the matrices $\mathbf{S}_l$ must be inserted in places of the scalar coefficients $s_l$ of a single-step response, $l = 1, 2, ..., D$. The formula (3.12), basic for the step response modeling, takes then the following vector-matrix form

$$y(k) = y(0) + \sum_{j=1}^{k} \mathbf{S}_j \triangle u(k-j) \tag{3.20}$$

where the process outputs and the process inputs changes are now vectors $y(k) \in \mathbb{R}^{n_y}$, $\triangle u(k-j) \in \mathbb{R}^{n_u}$. Analogously, the formula describing the predicted outputs becomes

$$y(k+p|k) = \sum_{j=1}^{p} \mathbf{S}_j \triangle u(k+p-j|k) + y(k) + \sum_{j=1}^{D-1} (\mathbf{S}_{j+p} - \mathbf{S}_j) \triangle u(k-j) \tag{3.21}$$

Thus, the formulae for elements of free and forced trajectories of predicted outputs have the following vector-matrix forms

$$\triangle y(k+p|k) = \sum_{j=1}^{p} \mathbf{S}_j \, \triangle u(k+p-j|k), \quad p = 1, 2, ..., N \tag{3.22}$$

$$y^0(k+p|k) = y(k) + \sum_{j=1}^{D-1} (\mathbf{S}_{j+p} - \mathbf{S}_j) \triangle u(k-j), \quad p = 1, 2, ..., N \tag{3.23}$$

### 3.2.2 Unconstrained Explicit DMC Algorithm

Let us define vectors

$$\mathcal{Y}^{sp}(k) = \begin{bmatrix} y^{sp}(k+N_1|k) \\ \vdots \\ y^{sp}(k+N|k) \end{bmatrix}, \quad \mathcal{Y}^0(k) = \begin{bmatrix} y^0(k+N_1|k) \\ \vdots \\ y^0(k+N|k) \end{bmatrix}$$

$$\Delta\mathcal{Y}(k) = \begin{bmatrix} \triangle y(k+N_1|k) \\ \vdots \\ \triangle y(k+N|k) \end{bmatrix}, \quad \Delta\mathcal{U}(k) = \begin{bmatrix} \triangle u(k|k) \\ \vdots \\ \triangle u(k+N_u-1|k) \end{bmatrix}$$

$$\mathcal{Y}^{pred}(k) = \mathcal{Y}^0(k) + \Delta\mathcal{Y}(k) = [\, y(k+N_1|k)^T \cdots \; y(k+N|k)^T \,]^T$$

and matrices

$$\underline{\mathbf{\Psi}} = \begin{bmatrix} \mathbf{\Psi}(N_1) & 0 & \cdots & 0 \\ 0 & \mathbf{\Psi}(N_1+1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{\Psi}(N) \end{bmatrix} \tag{3.24}$$

$$\underline{\mathbf{\Lambda}} = \begin{bmatrix} \mathbf{\Lambda}(0) & 0 & \cdots & 0 \\ 0 & \mathbf{\Lambda}(1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{\Lambda}(N_u-1) \end{bmatrix} \tag{3.25}$$

Then the cost function (3.4) can be written in the following form

$$J(k) = \left\| [\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] - \Delta\mathcal{Y}(k) \right\|^2_{\underline{\mathbf{\Psi}}} + \|\Delta\mathcal{U}(k)\|^2_{\underline{\mathbf{\Lambda}}} \tag{3.26}$$

If, however, additionally $\underline{\mathbf{\Psi}}= \mathbf{I}$ and $\underline{\mathbf{\Lambda}}=\lambda\mathbf{I}$, like in (3.5), then the cost function takes the form

$$J(k) = \left\| [\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] - \Delta\mathcal{Y}(k) \right\|^2 + \lambda \left\| \Delta\mathcal{U}(k) \right\|^2 \tag{3.27}$$

Additionally, let us define vectors

$$\mathcal{Y}(k) = \begin{bmatrix} y(k) \\ \vdots \\ y(k) \end{bmatrix}, \quad \Delta\mathcal{U}^P(k) = \begin{bmatrix} \triangle u(k-1) \\ \vdots \\ \triangle u(k-(D-1)) \end{bmatrix}$$

where $\dim \mathcal{Y}(k) = n_{\mathcal{Y}} = n_y \cdot (N - N_1 + 1)$, $\dim \Delta\mathcal{U}^P(k) = n_u \cdot (D-1)$.

Let us assume initially, for notation simplicity, that the *control process is one-dimensional* (SISO), *i.e.*, $n_y = n_u = 1$. The vectors $\mathcal{Y}^0(k)$, $\mathcal{Y}(k)$ and $\Delta\mathcal{Y}(k)$ are then of a dimension $n_{\mathcal{Y}} = N-(N_1-1)$. These vectors are evaluated in the DMC algorithm on the basis of a process model in a form of a finite step response $\{s_l,\ l = 1, 2, ..., D,\ s_l = s_D = k_m \text{ for } l > D\}$. This means that asymptotically stable processes are considered. Using (3.17) we can write the formula describing a *free component of the predicted output trajectory* in the following form

$$\mathcal{Y}^0(k) = \mathcal{Y}(k) + \mathbf{M}^P \Delta\mathcal{U}^P(k) \tag{3.28}$$

where the matrix $\mathbf{M}^P$ is of dimension $n_{\mathcal{Y}} \times (D-1)$ and for $N_1 = 1$ takes a characteristic form

$$\mathbf{M}^P = \begin{bmatrix} s_2 - s_1 & s_3 - s_2 & s_4 - s_3 & \cdots & s_D - s_{D-1} \\ s_3 - s_1 & s_4 - s_2 & s_5 - s_3 & \cdots & s_{D+1} - s_{D-1} \\ s_4 - s_1 & s_5 - s_2 & s_6 - s_3 & \cdots & s_{D+2} - s_{D-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{N+1} - s_1 & s_{N+2} - s_2 & s_{N+3} - s_3 & \cdots & s_{N+D-1} - s_{D-1} \end{bmatrix} \tag{3.29}$$

The general from for $N_1 > 1$ is obtained by dropping the first $N_1 - 1$ rows, *i.e.*,

$$\mathbf{M}^P = \begin{bmatrix} s_{1+N_1} - s_1 & s_{2+N_1} - s_2 & s_{3+N_1} - s_3 & \cdots & s_{D-1+N_1} - s_{D-1} \\ s_{2+N_1} - s_1 & s_{3+N_1} - s_2 & s_{4+N_1} - s_3 & \cdots & s_{D+N_1} - s_{D-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{N+1} - s_1 & s_{N+2} - s_2 & s_{N+3} - s_3 & \cdots & s_{N+D-1} - s_{D-1} \end{bmatrix} \tag{3.30}$$

The superscript "$P$" of this matrix was introduced to show that it is used for calculations of the output predictions depending only on past ("$P$" – "*Past*") increments of the process control input. Of course, $s_l = s_\infty = k_m$ for each element $s_l$ with the index $l \geq D$. For example, for $N = D = 6$ and $N_1 = 2$ (which is a standard value for the delay $\tau = 1$, *i.e.*, equal to one sampling period $T_p$) we obtain

$$\mathbf{M}^P = \begin{bmatrix} s_3 - s_1 & s_4 - s_2 & s_5 - s_3 & k_m - s_4 & k_m - s_5 \\ s_4 - s_1 & s_5 - s_2 & k_m - s_3 & k_m - s_4 & k_m - s_5 \\ s_5 - s_1 & k_m - s_2 & k_m - s_3 & k_m - s_4 & k_m - s_5 \\ k_m - s_1 & k_m - s_2 & k_m - s_3 & k_m - s_4 & k_m - s_5 \\ k_m - s_1 & k_m - s_2 & k_m - s_3 & k_m - s_4 & k_m - s_5 \end{bmatrix} \tag{3.31}$$

On the other hand, using (3.15) we can write *the forced component of the predicted output trajectory* in the following form

$$\Delta\mathcal{Y}(k) = \mathbf{M}\,\Delta\mathcal{U}(k) \tag{3.32}$$

where $\mathbf{M}$ is called the *dynamic matrix*, it is of dimension $n_\mathcal{Y} \times n_{\Delta\mathcal{U}}$ (for a SISO process, $n_\mathcal{Y} = N - N_1 + 1$, $n_{\Delta\mathcal{U}} = N_u$) and has the following form

$$\mathbf{M} = \begin{bmatrix} s_{N_1} & s_{N_1-1} & \cdots & s_1 & 0 & \cdots & 0 \\ s_{N_1+1} & s_{N_1} & \cdots & s_2 & s_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{N_u} & s_{N_u-1} & \cdots & s_{N_u-N_1+1} & s_{N_u-N_1} & \cdots & s_1 \\ s_{N_u+1} & s_{N_u} & \cdots & s_{N_u-N_1+2} & s_{N_u-N_1+1} & \cdots & s_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ s_N & s_{N-1} & \cdots & s_{N-N_1+1} & s_{N-N_1} & \cdots & s_{N-N_u+1} \end{bmatrix} \tag{3.33}$$

For example, for $N = 6$, $N_u = 3$ and $N_1 = \tau + 1 = 2$, we have

$$\mathbf{M} = \begin{bmatrix} s_2 & 0 & 0 \\ s_3 & s_2 & 0 \\ s_4 & s_3 & s_2 \\ s_5 & s_4 & s_3 \\ s_6 & s_5 & s_4 \end{bmatrix} \tag{3.34}$$

Generally, for the standard value $N_1 = \tau + 1$, where $\tau$ is the process time delay and, therefore, first $\tau = N_1 - 1$ coefficients of the step response are zeroes, the matrix $\mathbf{M}$ takes a characteristic form

$$\mathbf{M} = \begin{bmatrix} s_{N_1} & 0 & 0 & 0 & \cdots & 0 \\ s_{N_1+1} & s_{N_1} & 0 & 0 & \cdots & 0 \\ s_{N_1+2} & s_{N_1+1} & s_{N_1} & 0 & \cdots & 0 \\ s_{N_1+3} & s_{N_1+2} & s_{N_1+1} & s_{N_1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{N_u+N_1-1} & s_{N_u+N_1-2} & s_{N_u+N_1-3} & s_{N_u+N_1-4} & \cdots & s_{N_1} \\ s_{N_u+N_1} & s_{N_u+N_1-1} & s_{N_u+N_1-2} & s_{N_u+N_1-3} & \cdots & s_{N_1+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ s_N & s_{N-1} & s_{N-2} & s_{N-3} & \cdots & s_{N-N_u+1} \end{bmatrix} \tag{3.35}$$

Let us now consider the general case of a *multi-input multi-output (MIMO) process*. Using the model in the form of a matrix step response $\{\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_D\}$, we obtain equivalents of (3.28) and (3.32) in the same form

$$\mathcal{Y}^0(k) = \mathcal{Y}(k) + \mathbf{M}^P \Delta\mathcal{U}^P(k) \tag{3.36}$$

$$\Delta\mathcal{Y}(k) = \mathbf{M}\, \Delta\mathcal{U}(k) \tag{3.37}$$

only matrices $\mathbf{S}_j$ are now elements of the matrices $\mathbf{M}^P$ and $\mathbf{M}$, instead of scalars $s_j$. Thus we have

$$\mathbf{M}^P = \begin{bmatrix} \mathbf{S}_{1+N_1} - \mathbf{S}_1 & \mathbf{S}_{2+N_1} - \mathbf{S}_2 & \cdots & \mathbf{S}_{D-1+N_1} - \mathbf{S}_{D-1} \\ \mathbf{S}_{2+N_1} - \mathbf{S}_1 & \mathbf{S}_{3+N_1} - \mathbf{S}_2 & \cdots & \mathbf{S}_{D+N_1} - \mathbf{S}_{D-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{N+1} - \mathbf{S}_1 & \mathbf{S}_{N+2} - \mathbf{S}_2 & \cdots & \mathbf{S}_{N+D-1} - \mathbf{S}_{D-1} \end{bmatrix} \tag{3.38}$$

and

$$\mathbf{M} = \begin{bmatrix} \mathbf{S}_{N_1} & \mathbf{S}_{N_1-1} & \cdots & \mathbf{S}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{S}_{N_1+1} & \mathbf{S}_{N_1} & \cdots & \mathbf{S}_2 & \mathbf{S}_1 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{N_u} & \mathbf{S}_{N_u-1} & \cdots & \mathbf{S}_{N_u-N_1+1} & \mathbf{S}_{N_u-N_1} & \cdots & \mathbf{S}_1 \\ \mathbf{S}_{N_u+1} & \mathbf{S}_{N_u} & \cdots & \mathbf{S}_{N_u-N_1+2} & \mathbf{S}_{N_u-N_1+1} & \cdots & \mathbf{S}_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_N & \mathbf{S}_{N-1} & \cdots & \mathbf{S}_{N-N_1+1} & \mathbf{S}_{N-N_1} & \cdots & \mathbf{S}_{N-N_u+1} \end{bmatrix} \tag{3.39}$$

According to definitions of the vectors $\mathcal{Y}^0(k)$, $\Delta\mathcal{Y}(k)$, $\Delta\mathcal{U}^P(k)$ and $\Delta\mathcal{U}(k)$ the dimension of the matrix $\mathbf{M}^P$ is $n_{\mathcal{Y}} \times n_{\Delta\mathcal{U}^P} = n_y \cdot (N - N_1 + 1) \times n_u \cdot (D-1)$, while the dynamic matrix $\mathbf{M}$ is of dimension $n_{\mathcal{Y}} \times n_{\Delta\mathcal{U}} = n_y \cdot (N - N_1 + 1) \times n_u \cdot N_u$.

With the dynamic matrix $\mathbf{M}$, we can write the cost function (3.26) in the following form

$$J(k) = \left\| [\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] - \mathbf{M}\Delta\mathcal{U}(k) \right\|^2_{\underline{\mathbf{\Psi}}} + \|\Delta\mathcal{U}(k)\|^2_{\underline{\mathbf{\Lambda}}} \tag{3.40}$$

Under the assumed properties of the weighting matrices the cost function is strictly convex. Therefore, a necessary and sufficient condition of a minimum (without constraints) is zeroing of the gradient, *i.e.*,

$$-\mathbf{M}^T\underline{\mathbf{\Psi}}[\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k) - \mathbf{M}\Delta\mathcal{U}(k)] + \underline{\mathbf{\Lambda}}\Delta\mathcal{U}(k) = \mathbf{0} \tag{3.41}$$

$$[\mathbf{M}^T\underline{\mathbf{\Psi}}\mathbf{M}+\underline{\mathbf{\Lambda}}]\Delta\mathcal{U}(k) - \mathbf{M}^T\underline{\mathbf{\Psi}}[\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] = \mathbf{0} \tag{3.42}$$

From this equation *the vector of optimal control input increments* $\Delta\widehat{\mathcal{U}}(k)$ can be easily obtained

$$\begin{aligned}\Delta\widehat{\mathcal{U}}(k) &= [\mathbf{M}^T\underline{\mathbf{\Psi}}\mathbf{M}+\underline{\mathbf{\Lambda}}]^{-1}\mathbf{M}^T\underline{\mathbf{\Psi}}[\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] \\ &= \mathbf{K}[\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)]\end{aligned} \tag{3.43}$$

where

$$\mathbf{K} = [\mathbf{M}^T\underline{\mathbf{\Psi}}\mathbf{M}+\underline{\mathbf{\Lambda}}]^{-1}\mathbf{M}^T\underline{\mathbf{\Psi}} \tag{3.44}$$

is a matrix of dimension $n_{\triangle\mathcal{U}} \times n_{\mathcal{Y}}$. For the case $\underline{\mathbf{\Psi}} = \mathbf{I}$ and $\underline{\mathbf{\Lambda}} = \lambda\mathbf{I}$ this matrix takes the form

$$\mathbf{K} = [\mathbf{M}^T\mathbf{M}+\lambda\mathbf{I}]^{-1}\mathbf{M}^T \tag{3.45}$$

Let us denote

$$\mathbf{K} = \begin{bmatrix} \overline{\mathbf{K}}_1 \\ \overline{\mathbf{K}}_2 \\ \vdots \\ \overline{\mathbf{K}}_{N_u} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{1,1} & \mathbf{K}_{1,2} & \cdots & \mathbf{K}_{1,N-N_1+1} \\ \mathbf{K}_{2,1} & \mathbf{K}_{2,2} & \cdots & \mathbf{K}_{2,N-N_1+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{K}_{N_u,1} & \mathbf{K}_{N_u,2} & \cdots & \mathbf{K}_{N_u,N-N_1+1} \end{bmatrix} \tag{3.46}$$

where each sub-matrix $\overline{\mathbf{K}}_i$ is of dimension $n_u \times n_{\mathcal{Y}} = n_u \times n_y \cdot (N - N_1 + 1)$, and each sub-matrix $\mathbf{K}_{i,j}$ is of dimension $n_u \times n_y$.

In a predictive control algorithm only input increments evaluated for the current sampling instant $k$ are actually applied to the plant, *i.e.*, the optimal vector $\triangle\hat{u}(k|k)$ consisting of the first $n_u$ elements of the entire vector (3.43). We shall denote this value by $\triangle\hat{u}(k)$,

$$\triangle\hat{u}(k) = \triangle\hat{u}(k|k) = \overline{\mathbf{K}}_1[\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] \tag{3.47}$$

Therefore, the obtained *control law is a linear feedback from the difference between the set-point trajectory and the predicted free trajectory.*

We shall now present the DMC control law in a form showing a direct dependence of the optimal controller output on the past process inputs (*i.e.*, past controller outputs) and on the process outputs. Using (3.36) we have

$$\triangle\hat{u}(k) = \overline{\mathbf{K}}_1[\mathcal{Y}^{sp}(k) - \mathcal{Y}(k) - \mathbf{M}^P \triangle\mathcal{U}^P(k)]$$

Presenting the matrix $\mathbf{M}^P$ as

$$\mathbf{M}^P = \begin{bmatrix} \mathbf{M}_1^P & \mathbf{M}_2^P & \cdots & \mathbf{M}_{D-1}^P \end{bmatrix}$$

where dimension of each sub-matrix $\mathbf{M}_j^P$ is $n_{\mathcal{Y}} \times n_u = n_y \cdot (N - N_1 + 1) \times n_u$, and exploiting the structure (3.46) of the matrix $\mathbf{K}$ we can write

$$\begin{aligned} \triangle\hat{u}(k) &= \overline{\mathbf{K}}_1[\mathcal{Y}^{sp}(k) - \mathcal{Y}(k)] - \sum_{j=1}^{D-1} (\overline{\mathbf{K}}_1 \mathbf{M}_j^P) \triangle u(k-j) \\ &= \sum_{p=N_1}^{N} \mathbf{K}_{1,p-N_1+1}[y^{sp}(k+p|k) - y(k)] - \sum_{j=1}^{D-1} \mathbf{K}_j^u \triangle u(k-j) \end{aligned} \quad (3.48)$$

where

$$\mathbf{K}_j^u = \overline{\mathbf{K}}_1 \mathbf{M}_j^P, \quad j = 1, 2, ..., D-1 \quad (3.49)$$

Formula (3.48) presents the structure of the DMC control law designed analytically in the case without inequality constraints, or when the existence of these constraints has been consciously neglected during the design. Thus, this controller can be referred to as an *unconstrained, explicit DMC controller*, the description *analytical DMC controller* can also be met [134]. The structure of such a controller is illustrated in Fig. 3.4, where the box with a diagonal matrix of discrete transfer functions

$$\frac{1}{1 - z^{-1}}\mathbf{I} \quad (3.50)$$

stands for a discrete multivariable integration – a summation of consecutive increments of the process input vector in order to transform the control input increments into the control input values. Further, the box in the internal feedback loop fed by consecutive values of the controller output increments $\triangle u(k)$ should be treated as performing all which is necessary to implement the formula written in there, *i.e.*, also keeping in its memory $D-1$ last control input increments.

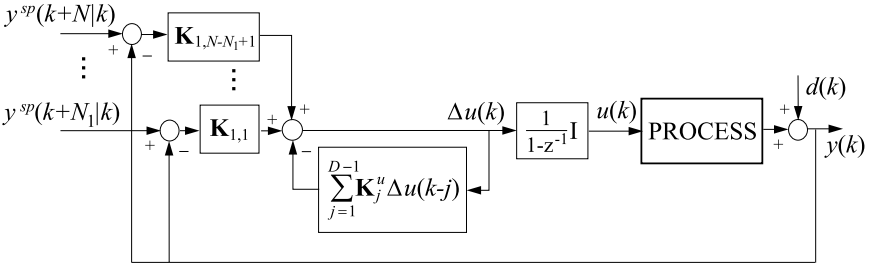


**Fig. 3.4.** Structure of the explicit, unconstrained DMC control law

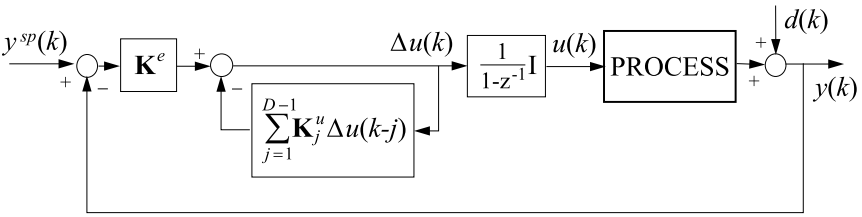


**Fig. 3.5.** Structure of the explicit, unconstrained DMC control law, for constant set-point trajectory over the prediction horizon

Let us note that the explicit, unconstrained DMC controller is *a linear controller* and matrices of its coefficients $\mathbf{K}_{1,p}$, $p = 1, ..., N - N_1 + 1$ and $\mathbf{K}_j^u$, $j = 1, ..., D - 1$ are calculated only once, during the design phase (off-line).

Performing stabilization tasks of the continuous process control, one usually does not know when changes in the set-point values will occur in the future, *i.e.*, in the prediction horizon. Therefore, it is a common practice in the design of the MPC algorithms for the stabilization tasks to assume that the set-points will be constant over the prediction horizon and equal to the current values

$$y^{sp}(k + N_1|k) = y^{sp}(k + N_1 + 1|k) = \cdots = y^{sp}(k + N|k) = y^{sp}(k) \tag{3.51}$$

Then the control law (3.48) can be simplified to the form

$$\begin{aligned} \triangle\hat{u}(k) &= \sum_{p=N_1}^{N} \mathbf{K}_{1,p-N_1+1}[y^{sp}(k) - y(k)] - \sum_{j=1}^{D-1} \mathbf{K}_j^u \triangle u(k-j) \\ &= \mathbf{K}^e[y^{sp}(k) - y(k)] - \sum_{j=1}^{D-1} \mathbf{K}_j^u \triangle u(k-j) \end{aligned} \tag{3.52}$$

where

$$\mathbf{K}^e = \sum_{p=N_1}^{N} \mathbf{K}_{1,p-N_1+1} \tag{3.53}$$

The structure of the linear control law (3.52) is illustrated in Fig. 3.5.

Let us rewrite (3.52) as follows

$$\triangle u(k) + \sum_{j=1}^{D-1} \mathbf{K}_j^u z^{-j} \triangle u(k) = \mathbf{K}^e e(k),$$

where $e(k) = y^{sp}(k) - y(k)$, while $z^{-1}$ denotes a unit time delay operator. Further, let us rewrite the above formula to the form

$$\triangle u(k) = [\mathbf{I} + \sum_{j=1}^{D-1} \mathbf{K}_j^u z^{-j}]^{-1} \mathbf{K}^e e(k)$$

In the case of a SISO control system, matrices $\mathbf{K}^e$ and $\mathbf{K}_j^u$ are scalars $k^e$ and $k_j^u$, thus the obtained formula can be written as a single transfer function of a discrete, one-dimensional controller (in an incremental form)

$$\frac{\triangle u(k)}{e(k)} = \frac{k^e}{1 + k_1^u z^{-1} + k_2^u z^{-2} + \cdots + k_{D-1}^u z^{-(D-1)}} \tag{3.54}$$

For a specific application, having a discrete linear process model and a DMC controller designed for it in the form (3.54), it is possible to investigate properties of the closed-loop control system, such as location of poles, stability margins, etc. – depending on chosen values of the controller parameters. This can give a significant insight into the basic properties of the DMC controller, primarily in the considered situation without constraints. It can be also significant for practically more important cases with constraints, although then the control system becomes nonlinear and a direct use of the indicated method for its analysis is not possible.

**Example 3.1**
Let us consider a simple single-input single-output process described by the model in the form of the following step response

| $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ |
|---|---|---|---|---|---|
| 0 | 0 | 0.2 | 0.5 | 0.6 | 0.62 |

(3.55)

where $D = 6$. Let us assume that $N = 6$, $N_u = 3$ and $N_1 = 3$ due to a delay by two sampling periods clearly visible in the step response.

Prediction of the process outputs dependent on the past control inputs, $\mathcal{Y}^0(k) = \mathcal{Y}(k) + \mathbf{M}^P \Delta\mathcal{U}^P(k)$, takes the form

$$\begin{bmatrix} y^0(k+3|k) \\ y^0(k+4|k) \\ y^0(k+5|k) \\ y^0(k+6|k) \end{bmatrix} = \begin{bmatrix} y(k) \\ y(k) \\ y(k) \\ y(k) \end{bmatrix} + \mathbf{M}^P \begin{bmatrix} \triangle u(k-1) \\ \triangle u(k-2) \\ \triangle u(k-3) \\ \triangle u(k-4) \\ \triangle u(k-5) \end{bmatrix}$$

where

$$\mathbf{M}^P = \begin{bmatrix} 0.5 & 0.6 & 0.42 & 0.12 & 0.02 \\ 0.6 & 0.62 & 0.42 & 0.12 & 0.02 \\ 0.62 & 0.62 & 0.42 & 0.12 & 0.02 \\ 0.62 & 0.62 & 0.42 & 0.12 & 0.02 \end{bmatrix} = \\ = \begin{bmatrix} \mathbf{M}_1^P & \mathbf{M}_2^P & \mathbf{M}_3^P & \mathbf{M}_4^P & \mathbf{M}_5^P \end{bmatrix}$$

The dynamic matrix takes the following form

$$\mathbf{M} = \begin{bmatrix} 0.2 & 0 & 0 \\ 0.5 & 0.2 & 0 \\ 0.6 & 0.5 & 0.2 \\ 0.62 & 0.6 & 0.5 \end{bmatrix} \tag{3.56}$$

Let us assume that $\underline{\mathbf{\Psi}}= \mathbf{I}$ and $\underline{\mathbf{\Lambda}}=\lambda\mathbf{I}$, then

$$\mathbf{M}^T\mathbf{M}+\lambda\mathbf{I} = \begin{bmatrix} 1.0344+\lambda & 0.772 & 0.43 \\ 0.772 & 0.65+\lambda & 0.4 \\ 0.43 & 0.4 & 0.29+\lambda \end{bmatrix}$$

Thus, for $\lambda = 0.01$ we obtain

$$\begin{aligned} \mathbf{K} &= (\mathbf{M}^T\mathbf{M}+\lambda\mathbf{I})^{-1}\mathbf{M}^T \\ &= \begin{bmatrix} 1.7171 & 1.5996 & -0.4519 & 0.0678 \\ -2.6932 & -0.9300 & 2.5508 & -0.6326 \\ 1.1297 & -1.0529 & -2.0868 & 2.4130 \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{K}}_1 \\ \overline{\mathbf{K}}_2 \\ \overline{\mathbf{K}}_3 \end{bmatrix} \end{aligned}$$

Assuming (3.51), *i.e.*, $y^{sp}(k+3|k) = \cdots = y^{sp}(k+6|k) = y^{sp}(k)$, the control law in the form (3.52) is as follows

$$\begin{aligned} \triangle\hat{u}(k) &= (\sum_{p=3}^{6} k_{1,p-2})(y^{sp}(k) - y(k)) - \sum_{j=1}^{5}(\overline{\mathbf{K}}_1\,\mathbf{M}_j^P)\triangle u(k-j) \\ &= k^e(y^{sp}(k) - y(k)) - \sum_{j=1}^{5} k_j^u \triangle u(k-j) \end{aligned}$$

where

$$k^e = 2.9327, \quad \mathbf{k}^u = \begin{bmatrix} 1.5802 & 1.7839 & 1.2317 & 0.3519 & 0.0587 \end{bmatrix}$$

For $\lambda = 0.05$ the values of the controller parameters are

$$k^e = 2.0787, \quad \mathbf{k}^u = \begin{bmatrix} 1.1622 & 1.2715 & 0.8730 & 0.2494 & 0.0416 \end{bmatrix}$$

and for $\lambda = 0.1$

$$k^e = 1.7201, \quad \mathbf{k}^u = \begin{bmatrix} 0.9774 & 1.0546 & 0.7225 & 0.2064 & 0.0344 \end{bmatrix}$$

Chosen trajectories of the process output and input obtained in the closed-loop control system with the designed DMC unconstrained controller are presented in Figures 3.6, 3.7, 3.8 and 3.9. Figures 3.6, 3.7 and 3.8 show how a decrease in the value of $\lambda$ accelerates changes in the output, by using higher amplitudes of the control input signal – and conversely, how an increase in the value of $\lambda$ smoothes the trajectory of the input signal slowing down changes of the output. This is important if overly high values of the control input signal are generated by the controller, exceeding transmission possibilities of the actuator and leading to a saturation of the actuator's output signal. This usually leads to a significant decrease of the control quality, due to the fact that the constraints have not been considered during the design of the controller. Such a case is presented in Fig. 3.9, where the dotted curve marks the signal at the controller output (integrated signal), and the continuous curve

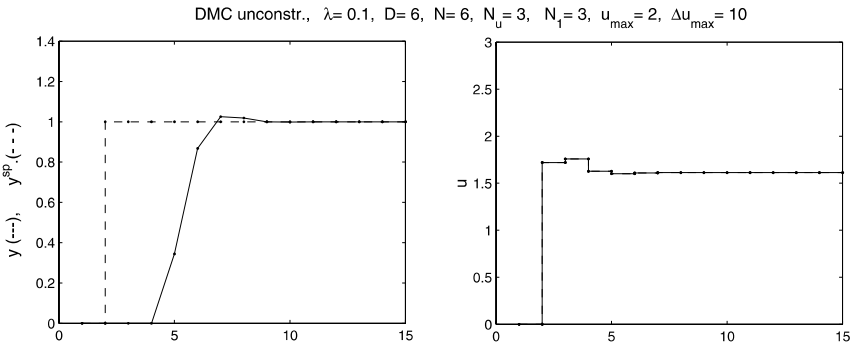


**Fig. 3.6.** Explicit controller, unconstrained trajectories, $\lambda = 0.1$

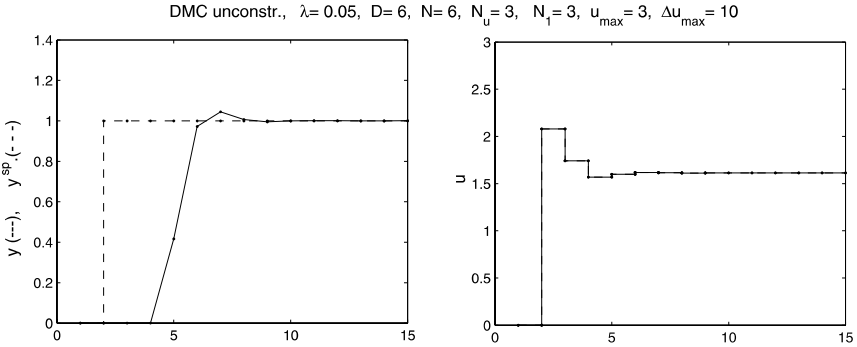


**Fig. 3.7.** Explicit controller, unconstrained trajectories, $\lambda = 0.05$

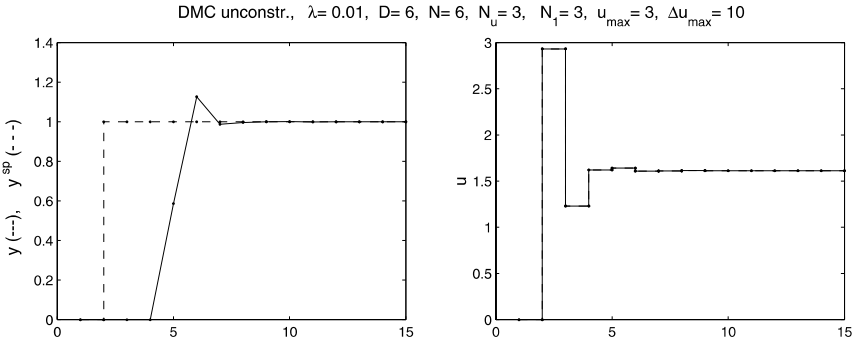


**Fig. 3.8.** Explicit controller, unconstrained trajectories, $\lambda = 0.01$

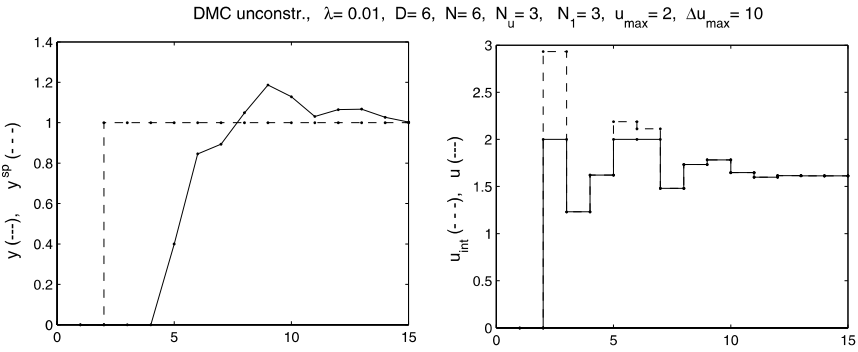


**Fig. 3.9.** Explicit controller, trajectories with active constraints on the controller output amplitude ignored during the controller structure design, $\lambda = 0.01$

marks the input signal actually affecting the process after passing through a nonlinear actuator constraining the amplitude of the process input signal to a value $u_{\max} = 2$ (compare with Fig. 3.8).

Moreover, the controllers designed with higher values of $\lambda$ are usually more robust against inaccuracies in process modeling or changes of plant parameters during the operation of the control system. □

It should be strongly emphasized that the optimal solution (3.43) to the controller unconstrained optimization problem, and hence the coefficients of the resulting control law, should not be obtained by a direct calculation of the inverse of the matrix $\mathbf{M}^T\underline{\mathbf{\Psi}}\mathbf{M}+\underline{\mathbf{\Lambda}}$ (by directly inverting this matrix). The reason is that this matrix can easily be ill-conditioned, especially for smaller values of diagonal entries of $\underline{\mathbf{\Lambda}}$, see *e.g.*, [82, 74].

According to state of the art in numerical analysis, if a desired solution can be obtained by solving a set of linear equations instead of by an inversion of a matrix, it should always be done in the former way for two reasons: the resulting solution is then more stable (smaller numerical errors) and more efficiently computed (less elementary mathematical operations). Therefore, if only the vector of optimal input increments is required, then it could be calculated as a solution to the set of linear equations (see (3.42))

$$[\mathbf{M}^T\underline{\mathbf{\Psi}}\mathbf{M}+\underline{\mathbf{\Lambda}}]\Delta\mathcal{U}(k) = \mathbf{M}^T\underline{\mathbf{\Psi}}[\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] \tag{3.57}$$

using an appropriate numerical method. However, this may not be the best and sufficient solution, since the reason of ill-conditioning of the matrix $\mathbf{M}^T\underline{\mathbf{\Psi}}\mathbf{M}$ is a possible ill-conditioning (or even rank-deficiency) of $\mathbf{M}$, especially for multivariable problems. Therefore, squaring up $\mathbf{M}$, *i.e.*, directly computing $\mathbf{M}^T\underline{\mathbf{\Psi}}\mathbf{M}$ should then be avoided.

A way to overcome the discussed difficulty is to treat the controller optimization problem as a least-squares problem, as indicated *e.g.*, in [82]. Since

matrices $\underline{\mathbf{\Psi}}$ and $\underline{\mathbf{\Lambda}}$ are positive semi-definite, as diagonal with all non-negative entries, then they can be immediately represented in a squared form

$$\underline{\mathbf{\Psi}} = \mathbf{S}_{\Psi}^{T}\mathbf{S}_{\Psi} \tag{3.58}$$

$$\underline{\mathbf{\Lambda}} = \mathbf{S}_{\Lambda}^{T}\mathbf{S}_{\Lambda} \tag{3.59}$$

Recall that the diagonal matrix $\underline{\mathbf{\Lambda}}$ is positive definite if all input increments over the control horizon are penalized, and the diagonal matrix $\underline{\mathbf{\Psi}}$ is positive definite when all control errors over the prediction horizon enter the objective function – else these matrices are positive-semidefinite.

Now, the objective function (3.40) can be written in the following form

$$J(k) = \left\| \begin{bmatrix} \mathbf{S}_{\Psi}(\mathcal{Y}^{sp}(k) - \mathbf{M}\Delta\mathcal{U}(k) - \mathcal{Y}^{0}(k)) \\ \mathbf{S}_{\Lambda}\Delta\mathcal{U}(k) \end{bmatrix} \right\|^{2} \tag{3.60}$$

The optimal solution to the controller unconstrained optimization problem can now be obtained as a least-squares solution to the vector equation

$$\begin{bmatrix} \mathbf{S}_{\Psi}(\mathcal{Y}^{sp}(k) - \mathbf{M}\Delta\mathcal{U}(k) - \mathcal{Y}^{0}(k)) \\ \mathbf{S}_{\Lambda}\Delta\mathcal{U}(k) \end{bmatrix} = \mathbf{0} \tag{3.61}$$

or written in the form

$$\begin{bmatrix} \mathbf{S}_{\Psi}\mathbf{M} \\ \mathbf{S}_{\Lambda} \end{bmatrix} \Delta\mathcal{U}(k) = \begin{bmatrix} \mathbf{S}_{\Psi}(\mathcal{Y}^{sp}(k) - \mathcal{Y}^{0}(k)) \\ \mathbf{0} \end{bmatrix} \tag{3.62}$$

The solution can be evaluated as

$$\Delta\mathcal{U}(k) = \mathbf{P} \begin{bmatrix} \mathbf{S}_{\Psi}(\mathcal{Y}^{sp}(k) - \mathcal{Y}^{0}(k)) \\ \mathbf{0} \end{bmatrix} \tag{3.63}$$

where $\mathbf{0}$ is a vector of length $n_u \cdot N_u$, and the matrix $\mathbf{P}$ is calculated as a Moore-Penrose pseudoinverse, using SVD decomposition $\mathbf{V\Sigma U}^{T}$ of the matrix $\begin{bmatrix} \mathbf{S}_{\Psi}\mathbf{M} \\ \mathbf{S}_{\Lambda} \end{bmatrix}$, *i.e.*,

$$\mathbf{P} = \begin{bmatrix} \mathbf{S}_{\Psi}\mathbf{M} \\ \mathbf{S}_{\Lambda} \end{bmatrix}^{+} = \mathbf{U}\mathbf{\Sigma}^{-1}\mathbf{V}^{T} = [\ \mathbf{P}_1\ \mathbf{P}_2\ ] \tag{3.64}$$

where matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ are of dimensions $n_u \cdot N_u \times n_y \cdot (N - N_1 + 1)$ and $n_u \cdot N_u \times n_u \cdot N_u$, respectively (for SVD decomposition, see *e.g.*, [51]).

Then not only the optimal input vector is computed, but also the controller gain matrix $\mathbf{K}$ (3.44) can be more securely calculated as

$$\mathbf{K} = \mathbf{P}_1\mathbf{S}_{\Psi} \tag{3.65}$$

Remember that the unconstrained controller gain matrix is calculated only once, off-line during the design phase.

### 3.2.3 Constraining the Controller Output by Projection

In real control systems there always exist constraints on process input signals which result from physical limits of actuators. These constraints cannot be exceeded – consequently, they are known as *hard constraints*. Moreover, there can occur constraints on process output signals, which are usually of a technological nature and can physically be exceeded, although this may be risky and lead to such consequences as damage of the apparatus or incorrect product parameters. Therefore, these constraints can generally be treated as *soft constraints*, namely those whose temporary violation is possible. In Section 3.1 constraints on process control inputs and controlled outputs were presented when describing the principle of predictive control (see (3.7), (3.8), (3.9)), the question will also be discussed in Section 3.6.2.

Classical controllers, especially the PID controllers, most commonly met in industrial applications, are designed (tuned) without consideration of constraints. Precisely, a controller is designed to operate only for small signal variations, in a sufficiently small vicinity of a working point. Nevertheless, in many practical situations the PID controllers also operate in quite a satisfactory way in situations when the generated control signal exceeds constraints on its possible amplitude or rate of change, provided the basic structure of the PID control law is augmented in an appropriate way. Namely, a system preventing integration of the controller output signal is added, becoming active if this signal gets constrained, which is checked using a model of the actuator's nonlinearity or directly the measured signal from actuator's output (the so-called *anti-windup structures*), see *e.g.*, [52].

Deriving in the previous section the unconstrained, explicit control law of the DMC controller, in the form (3.48) or (3.52), we ignored the constraints – in this way such a formula could be efficiently derived analytically. However, there is the question of how such a control system with the unconstrained DMC control law would operate in situations when the generated controller output signal $\triangle\hat{u}(k)$ has a higher amplitude or rate of change than are possible to be transmitted by an actuator, *i.e.*, a signal which violates physical constraints of the actuator. It turns out that only if the structure of the unconstrained DMC control law is correctly complemented using a model of the actuator's constraints, then such a controller will operate in quite a satisfactory way in many situations, especially for processes with most commonly met, relatively simple dynamics [88].

Figure 3.10 presents an incorrect, wrong application of the unconstrained DMC control law in a situation when information about constraints on amplitudes of the process input signals is available. This structure is incorrect and can have severe negative consequences for the control quality. The main reason is that the controller internal feedback loop is fed back with previous increments of the controller output signal as calculated by the DMC control law, *i.e.*, with the values $\triangle\hat{u}(k-j)$ following from (3.48), and not with the increments $\triangle u(k-j)$ corresponding to the process input signals $u(k-j)$ which,

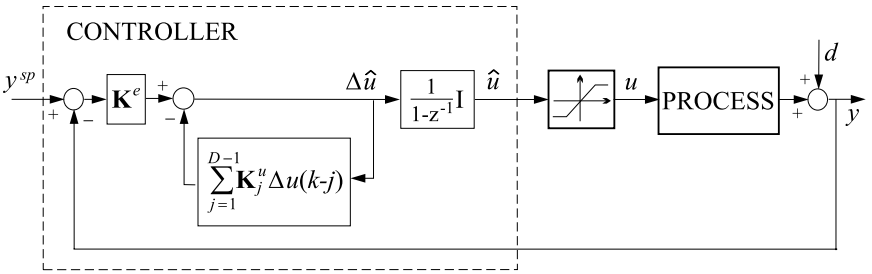


**Fig. 3.10.** Control structure with the explicit, unconstrained DMC controller ignoring constraints on process input signals – *a wrong structure*

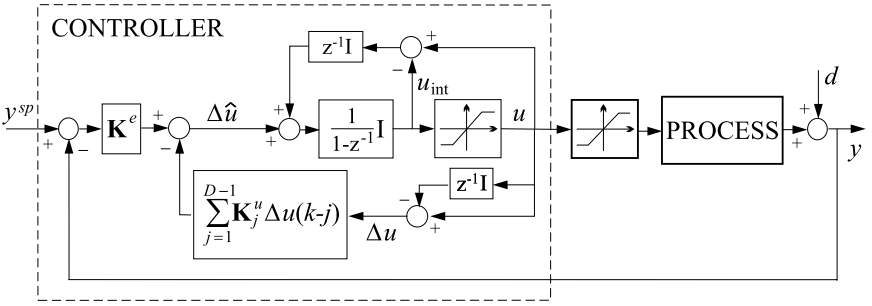


**Fig. 3.11.** Control structure with explicit, unconstrained DMC controller augmented by projection of its output onto constraints on the amplitudes of the process input signals, and by additional anti-windup loop

after passing through the constraining elements, really affected the process. The effects of the application of such an incorrect structure for a SISO system are presented in Figure 3.9 (within Example 3.1).

A correct control structure is presented in Fig. 3.11, where $z^{-1}\mathbf{I}$ denotes a unit delay (a delay by one sampling period) of the controller output/process input vector, while the nonlinear block should also be understood as a vector of $n_u = \dim u$ constraints on amplitudes of individual components of the process input. This structure contains also an additional feedback loop constraining the controller signal integration (correcting the state of the integrator) in a way which is similar to an anti-windup scheme used in structures of the PID controller, see *e.g.*, [3, 52]. Applying the additional anti-windup loop introduces *full correction of the state of the integrator*, with a unit delay – at a sampling instant $k$ the controller output signal $\triangle\hat{u}(k)$ is added to the signal $u(k-1)$ which *actually affected the process* at the previous sampling instant, *i.e.*, to the signal $u_{\text{int}}(k-1)$ after passing through the amplitude constraining element. An algorithmic realization of the presented structure, for a SISO

process control loop, is given below:

$$\begin{aligned}
&\triangle\hat{u}(k) = k^e(y^{sp}(k) - y(k)) - \textstyle\sum_{j=1}^{D-1} k_j^u \triangle u(k-j)\\
&u_{\mathrm{int}}(k) = u(k-1) + \triangle\hat{u}(k)\\
&\texttt{if } u_{\mathrm{int}}(k) \geq u_{\max}\\
&\qquad u(k) = u_{\max}\\
&\texttt{else } u(k) = \max\{u_{\mathrm{int}}(k), u_{\min}\}\\
&\texttt{end}\\
&\triangle u(k) = u(k) - u(k-1)
\end{aligned} \tag{3.66}$$

Functionally, the above algorithm performs a *projection* of the controller output signal onto the admissible set

$$u_{\min} \leq u(k) \leq u_{\max} \tag{3.67}$$

its action can also be described as *cutting off* the controller output signal.

In the structure shown in Fig. 3.11 and in the corresponding algorithm (3.66) values of the previous increments $\triangle u(k-j)$ used in the DMC feedback loop are those which have been modified in a correct way by passing through the constraints of the actuator. Thus, the general rule of a design of control structures implementing process input constraints is preserved: *past signals influencing the controller state should be the constrained signals, actually applied to the controlled plant*, see [52]. Fig. 3.12 presents trajectories obtained for an explicit DMC controller from Example 3.1 implemented in the structure from Fig. 3.11, but without the anti-windup feedback loop correcting the integrator state, while Fig. 3.13 presents analogous trajectories with this loop included. Comparison of these trajectories with those from Fig. 3.9 indicates significant improvement even in the case without the anti-windup loop, implementation of this loop further improves the controller operation.

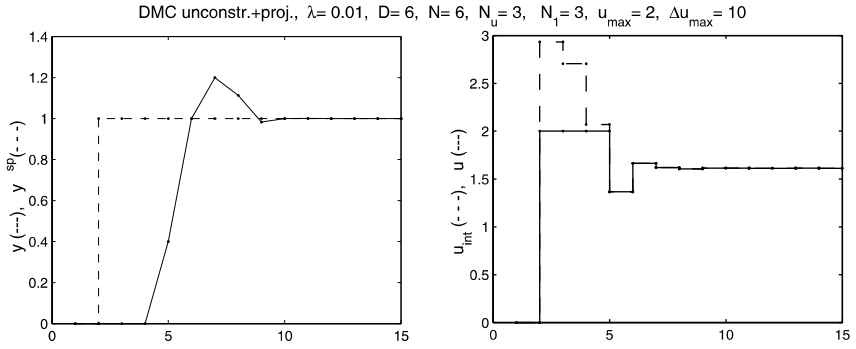


**Fig. 3.12.** Trajectories in the control system with the unconstrained DMC controller and projection onto the amplitude constraints, but without the anti-windup loop

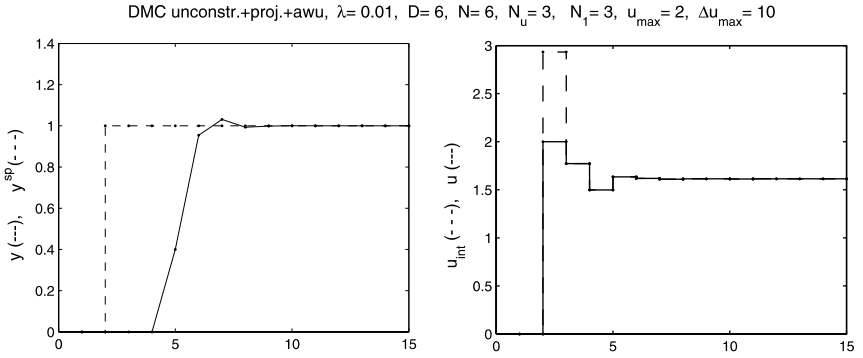


**Fig. 3.13.** Trajectories in the control system with the unconstrained DMC controller and projection onto the amplitude constraints, with the additional anti-windup loop

Except for constraints on process input amplitudes (3.67), there are also constraints on rates of change of the process input signals. As the DMC algorithm generates the process input signal increments, then these constraints are easy to satisfy. Fig. 3.14 presents the control structure taking into account constraints on both amplitudes and rates of change of the process input signals. The latter are realized by a nonlinear block located before the integrator, and saturation values in this block are the limiting values of the following inequality constraints

$$-\triangle u_{\max} \leq \triangle u(k) \leq \triangle u_{\max}$$

Figure 3.15 presents trajectories of the input and output variables obtained for the process from Example 3.1, in the control system presented in Fig. 3.10, *i.e.*, with an analytical DMC controller which does not use information about

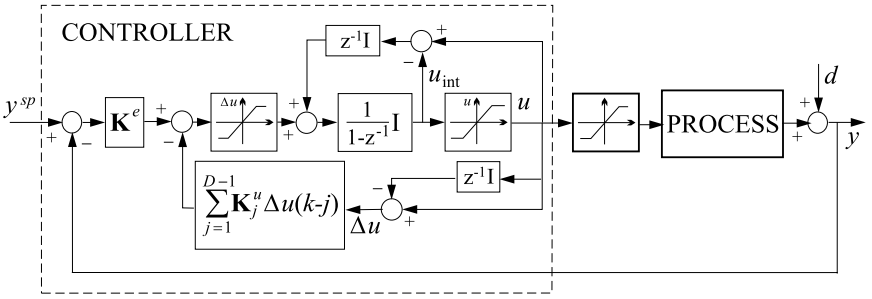


**Fig. 3.14.** Control structure with the unconstrained DMC controller and its output projected onto the constraints on amplitudes and rates of change of the process inputs, with the additional anti-windup loop

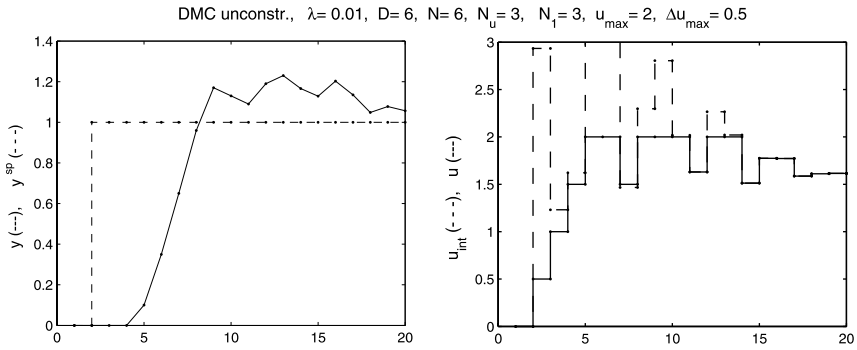


**Fig. 3.15.** Trajectories in the control system with the unconstrained DMC algorithm, with active amplitude ($\leq 2$) and rate of change ($\leq 0.5$) constraints not taken into account in the controller structure

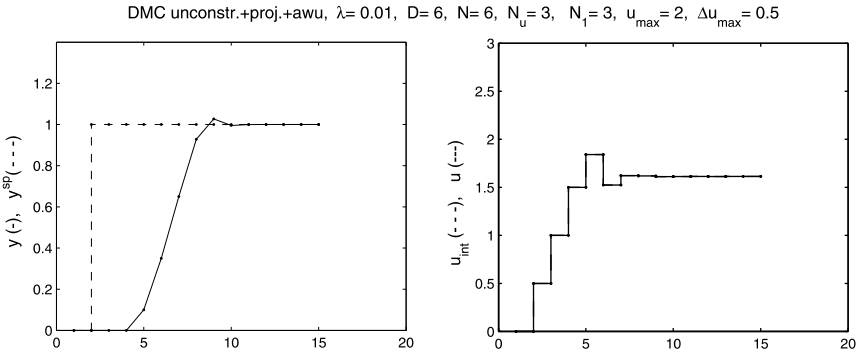


**Fig. 3.16.** Trajectories in the control system with the unconstrained DMC algorithm, with its output projected onto active constraints on the amplitude and rate of change of the process inputs, with the additional anti-windup loop

constraints – but in a situation when the actuator enforces active constraints (saturations) on amplitude and rate of change of process inputs. On the other hand, Fig. 3.16 presents trajectories obtained for identical conditions, but with the DMC controller taking into account the constraints, in the control structure from Fig. 3.14. It is worth to compare these trajectories with those presented in Fig. 3.12 and in Fig. 3.13. The differences are caused by the added constraint on the rate of change of the process input signal, $\triangle u_{\max} = 0.5$.

### 3.2.4 DMC Algorithm in Numerical Version

Treating the constraints on the controller output signal (*i.e.*, on the process input variable) as presented in the previous section is generally suboptimal, though often leads to results close to optimal. Moreover, it is much more

difficult to take into account in this way constraints on the process outputs (controlled and/or uncontrolled) – it is then necessary to make a transformation of these constraints into equivalent constraints on corresponding process inputs, using the process model, see [90].

The general principle of model predictive control was presented in Section 3.1. Its central element is a calculation of the process input signal as a solution of the constrained optimization problem (3.11), at each sampling instant. For a linear process model and without constraints, this problem has an explicit, analytical solution, which largely simplifies the implementation of the algorithm, as it was shown for the DMC algorithm. However, a great advantage from using the general formulation of the predictive control algorithm, with a numerical solution of a constrained optimization problem at each step, is that the constraints on process inputs and outputs can be taken into account in a direct and optimal way. If the process model is linear and so are the constraints, then the optimization problem is a *quadratic programming* (QP) problem. For such problems there exist effective and reliable optimization procedures which enable efficient implementations of the predictive control algorithms.

In the DMC algorithm the process model is given in the form of a step response, the controller cost function for such a model has already been formulated in the form (3.40). There remains only to formulate the constraints in an adequate way, as functions of the vector of decision variables, $\Delta\mathcal{U}(k)$. It is most simple for constraints on rate of change,

$$-\triangle u_{\max} \leq \triangle u(k+p|k) \leq \triangle u_{\max}, \quad p = 0, 1, ..., N_u - 1$$

Defining the vector $\Delta\mathcal{U}_{\max}$ of dimensionality $n_u \cdot N_u$,

$$\Delta\mathcal{U}_{\max} = \begin{bmatrix} \triangle u_{\max} \\ \vdots \\ \triangle u_{\max} \end{bmatrix}$$

we can write the constraints on rate of change in the following form

$$-\Delta\mathcal{U}_{\max} \leq \Delta\mathcal{U}(k) \leq \Delta\mathcal{U}_{\max}$$

The procedure is slightly more complicated for the amplitude constraints,

$$u_{\min} \leq u(k+p|k) \leq u_{\max}, \quad p = 0, 1, ..., N_u - 1$$

To this end, let us define first the following vectors of lower and upper bounds, of dimensionality $n_u \cdot N_u$,

$$\mathcal{U}_{\min} = \begin{bmatrix} u_{\min} \\ \vdots \\ u_{\min} \end{bmatrix}, \quad \mathcal{U}_{\max} = \begin{bmatrix} u_{\max} \\ \vdots \\ u_{\max} \end{bmatrix}$$

Then it remains to express values of the signals by their increments. Applying the equations

$$u(k+p|k) = u(k-1) + \sum_{j=0}^{p} \triangle u(k+j|k), \quad p = 0, 1, ..., N_u - 1$$

we can write the constraints on amplitude values in the following form

$$\mathcal{U}_{\min} \leq \mathcal{U}(k-1) + \mathbf{J}\Delta\mathcal{U}(k) \leq \mathcal{U}_{\max}$$

where

$$\mathcal{U}(k-1) = \begin{bmatrix} u(k-1) \\ u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix}, \quad \mathbf{J} = \begin{bmatrix} \mathbf{I}_u & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{I}_u & \mathbf{I}_u & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{I}_u & \mathbf{I}_u & \mathbf{I}_u & \mathbf{I}_u \end{bmatrix}$$

and the length of $\mathcal{U}(k-1)$ is $N_u \cdot n_u$, whereas every identity matrix $\mathbf{I}_u$ is of dimensionality $n_u \times n_u$. Finally, it is necessary to formulate correctly the constraints on process outputs (3.9)

$$y_{\min} \leq y(k+p|k) \leq y_{\max}, \quad p = N_1, N_1 + 1, ..., N \tag{3.68}$$

Defining, analogously to the vectors $\mathcal{U}_{\min}$ and $\mathcal{U}_{\max}$, the vectors $\mathcal{Y}_{\min}$ and $\mathcal{Y}_{\max}$, each of dimensionality $n_y \cdot (N - N_1 + 1)$, and applying (3.37), the output constraints can be written in the following form

$$\mathcal{Y}_{\min} \leq \mathcal{Y}^0(k) + \mathbf{M}\Delta\mathcal{U}(k) \leq \mathcal{Y}_{\max}$$

where $\mathcal{Y}^0(k)$ is given by (3.36).

In conclusion, at every sampling instant the following *quadratic optimization problem* of the *DMC algorithm in a numerical version* (*numerical DMC algorithm*) is solved:

$$\begin{aligned} \min_{\Delta\mathcal{U}(k)} \{ & \left\| [\mathcal{Y}^{sp}(k) - \mathcal{Y}(k) - \mathbf{M}^P \Delta\mathcal{U}^P(k)] - \mathbf{M}\Delta\mathcal{U}(k) \right\|^2_{\underline{\Psi}} + \|\Delta\mathcal{U}(k)\|^2_{\underline{\Lambda}} \} \\ \text{subj. to: } & -\Delta\mathcal{U}_{\max} \leq \Delta\mathcal{U}(k) \leq \Delta\mathcal{U}_{\max} \\ & \mathcal{U}_{\min} \leq \mathcal{U}(k-1) + \mathbf{J}\Delta\mathcal{U}(k) \leq \mathcal{U}_{\max} \\ & \mathcal{Y}_{\min} \leq \mathcal{Y}^0(k) + \mathbf{M}\Delta\mathcal{U}(k) \leq \mathcal{Y}_{\max} \end{aligned} \tag{3.69}$$

The problem (3.69) can easily be written in an equivalent form which is standard for quadratic programming, *e.g.*, in the form required by the QUADPROG (Quadratic Programming) procedure of the MATLAB® package

$$\begin{aligned} & \min\{J(x) = \frac{1}{2} x^T \mathbf{H} x + f^T x\} \\ & \text{subj. to: } x_{\min} \leq x \leq x_{\max} \\ & \mathbf{A}x \leq b \end{aligned} \tag{3.70}$$

It is easy to check that problems (3.69) and (3.70) are equivalent, if

$$x = \Delta\mathcal{U}(k), \quad x_{\min} = -\Delta\mathcal{U}_{\max}, \quad x_{\max} = \Delta\mathcal{U}_{\max}$$

$$\mathbf{H} = 2(\mathbf{M}^T\underline{\mathbf{\Psi}}\mathbf{M}+\underline{\mathbf{\Lambda}})$$

$$f = -2\mathbf{M}^T\underline{\mathbf{\Psi}}(\mathcal{Y}^{sp}(k) - \mathcal{Y}(k) - \mathbf{M}^P\Delta\mathcal{U}^P(k))$$

$$\mathbf{A} = \begin{bmatrix} -\mathbf{J} \\ \mathbf{J} \\ -\mathbf{M} \\ \mathbf{M} \end{bmatrix}, \quad b = \begin{bmatrix} -\mathcal{U}_{\min} + \mathcal{U}(k-1) \\ \mathcal{U}_{\max} - \mathcal{U}(k-1) \\ -\mathcal{Y}_{\min} + \mathcal{Y}^0(k) \\ \mathcal{Y}_{\max} - \mathcal{Y}^0(k) \end{bmatrix}$$

Currently, numerical algorithms from the group of active set methods and from the group of interior point methods are considered to be the most effective in solving the quadratic programming problems, see *e.g.*, [18, 1, 82, 115].

A numerical DMC controller, which calculates at each sampling instant the control input increment solving the presented quadratic programming problem, was also applied to control the process presented in Example 3.1, for the same operating conditions as those applied for the explicit DMC controller in the structure from Fig. 3.14. The obtained trajectories were identical as those obtained there and presented in Figures 3.13 and 3.16 – therefore we shall not quote them here. These facts confirm also the hypothesis that the explicit controller in the structure shown in Fig. 3.14 *works often well*, even in a way close to optimal. The author is not aware whether there exists a rigorous theoretical analysis indicating when the analytical controller in the structure from Fig. 3.14 operates as an (optimal) numerical controller.

As it was mentioned, the quadratic programming problem (3.69) can be precisely and effectively solved – however, on the condition that there is a solution to this problem, namely that the feasible set defined by all inequality constraints is not empty (that constraints are not contradictory). We have touched on this problem already in Section 3.1, when discussing general formulation of the control optimization problem in the predictive control algorithm with a finite receding horizon. The problem (3.69) is a special case of this general formulation and all comments made in Section 3.1 apply here. Let us remind here only that the feasible set can happen to be empty when there exist constraints on process outputs, (3.68). The first point of Section 3.6.2 is devoted to the question how to ensure that the feasible set is always not empty.

### 3.2.5 Model Uncertainty, Disturbances

A linear model used for prediction of the process output values in a DMC algorithm is a certain approximation of a real process input-output mapping, usually due to a *structural uncertainty* (the assumption of linearity) and a *parametric uncertainty* (inaccurate coefficient values). Moreover, a process is

usually affected by uncontrolled inputs, *i.e.*, *disturbances*. One can distinguish *measurable* (*measured*) and *unmeasurable* disturbances.

**Uncertainty, Unmeasurable Disturbances**

A typical way of proceeding is to treat prediction errors which result from inaccuracies in process modeling and influence of not measured disturbances together, as a combined effect of influence of certain unmeasurable disturbances. In the DMC algorithm these disturbances are treated as integrated white noises acting additively on the process outputs. At the current sampling instant $k$, the effect of their action $d(k)$ is calculated as a difference between the process output values currently measured, $y(k)$, and the output values predicted for the current sampling instant at the previous one, $y(k|k-1)$ – as it was described in Section 3.2.1 when defining the *constant output disturbance prediction model* in the DMC algorithm. This model is based on the assumption that the unmeasured disturbances can be treated as integrated zero-mean white noises, which results in optimal predictions constant in the entire prediction horizon and equal to the value $d(k)$ calculated at the current sampling instant.

The assumed disturbance model works well for situations typical for industrial process control applications. Moreover, it ensures *zero steady-state control errors* – just like classical feedback controllers *with integration*, PI or PID. This fact can easily be explained assuming that a control system with a DMC controller is asymptotically stable and there exists such a feasible process input value $u^{sp}$ that

$$y^{sp} = \mathbf{K}_p u^{sp}$$

where $y^{sp}$ denotes set-point values for the controlled outputs, and $\mathbf{K}_p$ the matrix of static gains of the process. Under constant set-points and disturbances, the asymptotically stable closed-loop control system always stabilizes at certain equilibrium point (steady-state point). Let us denote process input and output signals at this state by $u_\infty$ and $y_\infty$, of course these values satisfy

$$y_\infty = \mathbf{K}_p u_\infty$$

In the steady-state all increments of the controller outputs, *i.e.*, past (over a process dynamics horizon) and future optimal (over a prediction horizon) increments are zero, so at every sampling instant $k$ the optimal value of the cost function in (3.69) is

$$J(k)_{\text{opt}} = \|\mathcal{Y}^{sp}(k) - \mathcal{Y}(k)\|^2_{\underline{\mathbf{\Psi}}} = \sum_{p=N_1}^{N} \|y^{sp} - y_\infty\|^2_{\mathbf{\Psi}(p)}$$

Therefore, in a steady-state the equality $y_\infty = y^{sp}$ must be fulfilled, because otherwise at each sampling instant $k$ there would exist non-zero controller

output increments which would change the values $u_\infty$ towards $u^{sp}$ in order to decrease the value of the cost function (because, from the assumption, $y^{sp} = \mathbf{K}_p u^{sp}$)). This would obviously contradict the fact that optimal increments of the controller outputs are equal to zero in the steady-state.

In the figures presented so far, showing trajectories in the DMC control system for a process from Example 3.1, it can easily be seen that the control error tends to zero, after a step change of the set-point from zero to one. Fig. 3.17 presents trajectories for the same process, but with a change of an unmeasurable disturbance $d(k)$ starting in the second step of the simulation, where the influence of the disturbance on the process output was described by an autoregressive model in the form $d(k+1) = ad(k)+0.45$ for $k \geq 1$ ($d(k) = 0$ for $k \leq 1$), with $a = 0.2676$, the value corresponding to the autoregression coefficient of the process step response. Fig. 3.18 presents the same situation,

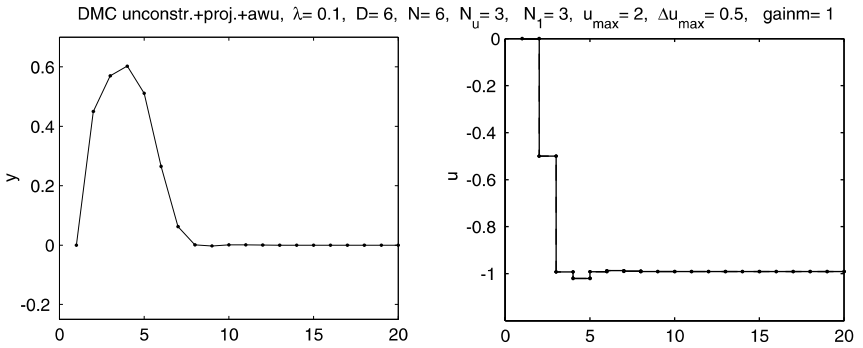


**Fig. 3.17.** Trajectories in the control system with the explicit DMC controller and constraints taken into account, after a change of the unmeasurable disturbance

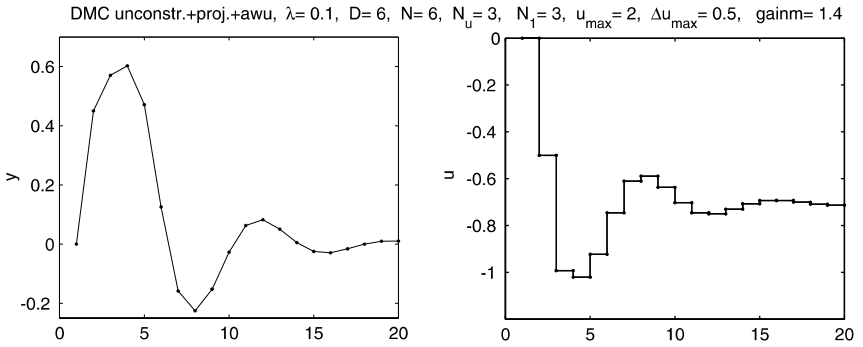


**Fig. 3.18.** Trajectories in the control system with the explicit DMC controller and constraints taken into account, after a change of the unmeasurable disturbance and with the process gain increased by 40% (leaving process model unchanged)

but with the process gain (in the control channel) increased by 40%, leaving the model unchanged. So, the controller had to deal additionally with large modeling errors. The trajectories are worse, as it could be expected, but the controller effectively zeroes the control error.

### Compensation of Measured Disturbances

When designing control systems the following well-known general rule should be followed: *the influence of measured disturbances*, *i.e.*, uncontrolled but measured process inputs, on the process outputs *should be compensated in an open-loop structure* – if only this influence is significant and dynamics of disturbance and control channels and available process models enable an efficient realization of the compensation. Control in the open-loop structure, called the *feedforward control*, is then faster and more effective than in a closed-loop structure. What remains for control in the feedback loop is a reduction of influences of the remaining disturbances and inaccuracy of the compensation.

The above design rule should of course also be obeyed in predictive control. For compensation of measured disturbances in an open-loop it is necessary to have a model of an influence of these variables on the controlled outputs. In a DMC algorithm this model should be created in the same form as the basic process model of the control channel, *i.e.*, in the form of step responses. Let us denote by $z$ the vector of measured disturbances, of dimensionality $n_z = \dim z$, and by $\{\mathbf{S}_l^z,\ l = 1, 2, 3, ...\}$ a sequence of matrix step responses of the process outputs on unit steps in $z$, analogously as $\{\mathbf{S}_l,\ l = 1, 2, 3, ...\}$ denotes a sequence of matrix step responses of the process outputs on steps in control inputs, as described in Section 3.2.1. According to the principle of superposition, the process model can then be written as an extension of (3.20)

$$y(k) = y(0) + \sum_{j=1}^{k} \mathbf{S}_j \triangle u(k-j) + \sum_{j=1}^{k} \mathbf{S}_j^z \triangle z(k-j) \tag{3.71}$$

where $\triangle z(k-j) = z(k-j) - z(k-j-1)$, $j = 1, ..., k$.

The prediction equations resulting from (3.71) depend on the disturbance inputs only if no increments of the control inputs are assumed, and can be derived identically as it was done for the control inputs, see Section 3.2.1. In this way we obtain the following dependence

$$\begin{aligned} y(k+p|k) = y(k) + \sum_{j=1}^{p-1} \mathbf{S}_j^z \triangle z(k+p-j|k) + \mathbf{S}_p^z \triangle z(k) + \\ + \sum_{j=p+1}^{D_z-1+p} \mathbf{S}_j^z \triangle z(k+p-j) - \sum_{j=1}^{D_z-1} \mathbf{S}_j^z \triangle z(k-j) \end{aligned}$$

and after a simple rearrangement

$$y(k+p|k) = \sum_{j=1}^{p-1} \mathbf{S}_j^z \triangle z(k+p-j|k) + \\ + y(k) + \mathbf{S}_p^z \triangle z(k) + \sum_{j=1}^{D_z-1} (\mathbf{S}_{j+p}^z - \mathbf{S}_j^z) \triangle z(k-j) \qquad (3.72)$$

where $D_z$ denotes the horizon of disturbance dynamics, *i.e.*, for $j \geq D_z$ $\mathbf{S}_j^z = \mathbf{S}_{D_z}^z$ is assumed. In the obtained formula, the first sum represents the influence of the predicted values of the measured disturbances, while the remaining components represent the influence of values which have been measured.

Joining the components from the second line of (3.72) with (3.23) we obtain a formula describing a *free component of the predicted trajectory* of the controlled outputs, dependent on past control input values and *past and present values* of the measured disturbances

$$\mathcal{Y}^0(k) = \mathcal{Y}(k) + \mathbf{M}^P \Delta \mathcal{U}^P(k) + \mathbf{M}^{zP} \Delta \mathcal{Z}^P(k) \qquad (3.73)$$

In (3.73), $\Delta \mathcal{Z}^P(k)$ is a vector of past disturbance increments,

$$\Delta \mathcal{Z}^P(k) = \begin{bmatrix} \triangle z(k) \\ \triangle z(k-1) \\ \vdots \\ \triangle z(k-(D_z-1)) \end{bmatrix}$$

while the matrix $\mathbf{M}^{zP}$ is given by the formula

$$\mathbf{M}^{zP} = \begin{bmatrix} \mathbf{S}_{N_1}^z & \mathbf{S}_{1+N_1}^z - \mathbf{S}_1^z & \mathbf{S}_{2+N_1}^z - \mathbf{S}_2^z & \cdots & \mathbf{S}_{D_z-1+N_1}^z - \mathbf{S}_{D_z-1}^z \\ \mathbf{S}_{N_1+1}^z & \mathbf{S}_{2+N_1}^z - \mathbf{S}_1^z & \mathbf{S}_{3+N_1}^z - \mathbf{S}_2^z & \cdots & \mathbf{S}_{D_z+N_1}^z - \mathbf{S}_{D_z-1}^z \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_N^z & \mathbf{S}_{N+1}^z - \mathbf{S}_1^z & \mathbf{S}_{N+2}^z - \mathbf{S}_2^z & \cdots & \mathbf{S}_{N+D_z-1}^z - \mathbf{S}_{D_z-1}^z \end{bmatrix} \qquad (3.74)$$

Let us note that structure of the matrix $\mathbf{M}^{zP}$ is similar to the structure of the matrix $\mathbf{M}^P$, compare with (3.38). The only difference is that $\mathbf{M}^{zP}$ is extended by the additional first column, corresponding to the first column of the matrix $\mathbf{M}$, compare with (3.39). This results from the fact that at sampling instant $k$ the vector $\triangle z(k)$ represents already realized (and measured) disturbance values, while $\triangle u(k|k)$ belongs to decision variables.

The predicted process output values are influenced not only by past control inputs and measured disturbance values, but also by actual and future control input values $u(k+p|k)$, $p = 0, 1, ..., N_u - 1$ calculated at sampling instant $k$ and by future values of disturbances $z(k+p|k)$, $p = 1, 2, ..., N-1$. The control inputs $u(k+p|k)$, $p = 0, 1, ..., N_u - 1$ are *decision variables*, while the future disturbance inputs $z(k+p|k)$ should be *forecasted* at sampling instant $k$, in an appropriate way. The simplest way is to assume that the

future disturbance values are equal to the current value in the prediction horizon, namely $z(k+p|k) = z(k),\ p = 1, ..., N-1$, therefore $\triangle z(k+p|k) = z(k+p|k) - z(k+p-1|k) = 0,\ p = 1, ..., N-1$. The above assumption has led to the disturbance model used in (3.73) which assumes a lack of information about possible future changes of the measured disturbances.

However, having in hand the mechanism of predictive control, it is possible to consider the future influence of measured disturbances in a more sophisticated way, if the trajectories of these disturbances indicate significant changes in time windows comparable to the control horizon or to the prediction horizon, and are of a continuous, smooth character. Then, using the gathered data it may be reasonable to construct a *disturbance predictor*, *e.g.*, a simple model of autoregressive (AR) structure with parameters calculated by one of standard identification methods. Let us note that most important for correct operation of a predictive controller is the disturbance prediction for the first part of the prediction horizon, as at each of the following sampling instants the entire process of control calculation will be repeated. Thus an AR model can be of a relatively low order. Moreover, for longer prediction horizons $N$ it may be reasonable to introduce a *disturbance horizon* $N_z < N$, *i.e.*, to assume that $z(k+p|k) = z(k+N_z|k),\ p = N_z+1, ..., N-1$.

Denoting by $z(k+p|k),\ p = 1, ..., N-1,$ future values of the measured disturbances, forecasted at sampling instant $k$ on the basis of their past values $z(k-j),\ j = 0, 1, ...$ by a linear disturbance predictor, let us define the vector

$$\Delta\mathcal{Z}(k) = \begin{bmatrix} \triangle z(k+1|k) \\ \vdots \\ \triangle z(k+N_z|k) \end{bmatrix}$$

Instead of (3.73), we can now use a formula which defines predicted values of the controlled outputs as depending on past control inputs, past measured disturbance values and *forecasted measured disturbance values*, as follows

$$\mathcal{Y}^0(k) = \mathcal{Y}(k) + \mathbf{M}^P \Delta U^P(k) + \mathbf{M}^{zP} \Delta\mathcal{Z}^P(k) + \mathbf{M}^z \Delta\mathcal{Z}(k), \tag{3.75}$$

where the matrix $\mathbf{M}^z$, of dimension $n_{\mathcal{Y}} \times n_{\triangle\mathcal{Z}} = n_y \cdot (N - N_1 + 1) \times n_z \cdot N_z$, is of the form

$$\mathbf{M}^z = \begin{bmatrix} \mathbf{S}^z_{N_1-1} & \mathbf{S}^z_{N_1-2} & \cdots & \mathbf{S}^z_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{S}^z_{N_1} & \mathbf{S}^z_{N_1-1} & \cdots & \mathbf{S}^z_2 & \mathbf{S}^z_1 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}^z_{N_z} & \mathbf{S}^z_{N_z-1} & \cdots & \mathbf{S}^z_{N_z-N_1+2} & \mathbf{S}^z_{N_z-N_1+1} & \cdots & \mathbf{S}^z_1 \\ \mathbf{S}^z_{N_z+1} & \mathbf{S}^z_{N_z} & \cdots & \mathbf{S}^z_{N_z-N_1+3} & \mathbf{S}^z_{N_z-N_1+2} & \cdots & \mathbf{S}^z_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}^z_{N-1} & \mathbf{S}^z_{N-2} & \cdots & \mathbf{S}^z_{N-N_1+1} & \mathbf{S}^z_{N-N_1} & \cdots & \mathbf{S}^z_{N-N_z} \end{bmatrix} \tag{3.76}$$

The structure of this matrix is similar to the structure of the dynamic matrix $\mathbf{M}$, compare with (3.39). The differences are due to the fact that vectors $\Delta\mathcal{Z}(k)$ and $\Delta\mathcal{U}(k)$ are shifted by one sampling period in relation to each other: the vector $\Delta\mathcal{Z}(k)$ consists of $N_z$ sub-vectors $\triangle z(k+p|k)$ indexed by $p = 1, 2, ..., N_z$, while the vector $\Delta\mathcal{U}(k)$ consists of $N_u$ sub-vectors $\triangle u(k+p|k)$ indexed by $p = 0, 1, ..., N_u - 1$.

Denote by $\mathbf{Z}$ a matrix representing a *linear disturbance predictor*, describing the relation between future and past disturbance values, *i.e.*,

$$\Delta\mathcal{Z}(k) = \mathbf{Z}\,\Delta\mathcal{Z}^P(k)$$

We can now transform (3.75) to the following form

$$\begin{aligned}\mathcal{Y}^0(k) &= \mathcal{Y}(k) + \mathbf{M}^P\Delta\mathcal{U}^P(k) + \mathbf{M}^{zP}\Delta\mathcal{Z}^P(k) + \mathbf{M}^z\mathbf{Z}\,\Delta\mathcal{Z}^P(k)\\ &= \mathcal{Y}(k) + \mathbf{M}^P\Delta\mathcal{U}^P(k) + \left[\mathbf{M}^{zP} + \mathbf{M}^z\mathbf{Z}\right]\Delta\mathcal{Z}^P(k)\\ &= \mathcal{Y}(k) + \mathbf{M}^P\Delta\mathcal{U}^P(k) + \mathbf{Z}^P\Delta\mathcal{Z}^P(k)\end{aligned} \tag{3.77}$$

where

$$\mathbf{Z}^P = \mathbf{M}^{zP} + \mathbf{M}^z\mathbf{Z} \tag{3.78}$$

Equality (3.77) defines the *free component of the predicted outputs trajectory* which is an extension of (3.36) to the case with measured disturbances. The entire reasoning leading to the derivation of the explicit unconstrained DMC control law presented in Section 3.2.2 can now be repeated, using only (3.77) instead of (3.36). That is why only the final result will now be presented, which corresponds to (3.47)

$$\triangle\hat{u}(k) = \overline{\mathbf{K}}_1[\mathcal{Y}^{sp}(k) - \mathcal{Y}(k)] - \overline{\mathbf{K}}_1\mathbf{M}^P\Delta\mathcal{U}^P(k) - \overline{\mathbf{K}}_1\mathbf{Z}^P\Delta\mathcal{Z}^P(k) \tag{3.79}$$

Partitioning now the matrix $\mathbf{Z}^P$ like we did previously with the matrix $\mathbf{M}^P$, as

$$\mathbf{Z}^P = [\mathbf{Z}_0^P\ \mathbf{Z}_1^P\ \mathbf{Z}_2^P\ \cdots\ \mathbf{Z}_{D_z-1}^P]$$

and using structure of the matrix $\mathbf{K}$ presented in (3.46), we obtain

$$\begin{aligned}\triangle\hat{u}(k) = &\sum_{p=N_1}^{N}\mathbf{K}_{1,p-N_1+1}[y^{sp}(k+p|k) - y(k)] - \sum_{j=1}^{D-1}(\overline{\mathbf{K}}_1\,\mathbf{M}_j^P)\triangle u(k-j) +\\ &- \sum_{j=0}^{D_z-1}(\overline{\mathbf{K}}_1\,\mathbf{Z}_j^P)\triangle z(k-j)\\ = &\sum_{p=N_1}^{N}\mathbf{K}_{1,p-N_1+1}[y^{sp}(k+p|k) - y(k)] - \sum_{j=1}^{D-1}\mathbf{K}_j^u\triangle u(k-j) +\\ &- \sum_{j=0}^{D_z-1}\mathbf{K}_j^z\triangle z(k-j)\end{aligned} \tag{3.80}$$

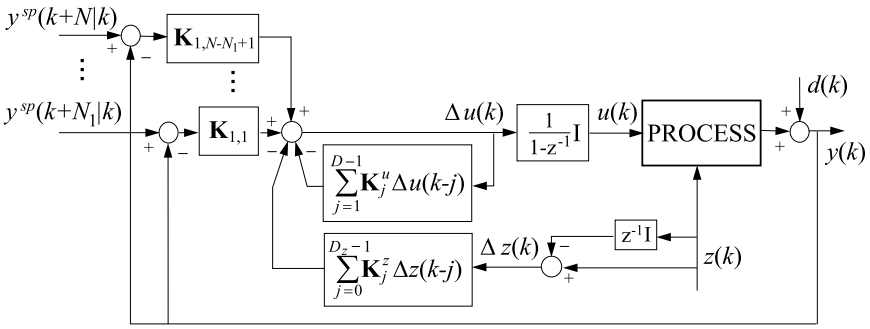


**Fig. 3.19.** Structure of the explicit DMC control law with feedforward compensation of measured disturbances

where

$$\mathbf{K}_j^z = \overline{\mathbf{K}}_1 \, \mathbf{Z}_j^P, \quad j = 0, 1, ..., D_z - 1$$

The formula (3.80) presents the structure of the explicit DMC control law which is an extension of (3.48), to the case with feedforward compensation of measured disturbances. The structure of the controller is illustrated in Fig. 3.19 (compare with Fig. 3.4 and see the comments there). Applying the explicit DMC algorithm with disturbance compensation for cases with constraints on the process input signals, this structure should be appropriately supplemented, analogously to the way it was presented in Fig. 3.11 and Fig. 3.14 for the structure without compensation of disturbances.

For the considered case with disturbance compensation the structure of the quadratic optimization problem of the *numerical DMC control algorithm* does not alter, only the cost function should be extended to

$$J(k) = \left\| [\mathcal{Y}^{sp}(k) - \mathcal{Y}(k) - \mathbf{M}^P \Delta\mathcal{U}^P(k) - \mathbf{Z}^P \Delta\mathcal{Z}^P(k)] - \mathbf{M}\Delta\mathcal{U}(k) \right\|_{\underline{\Psi}}^2 + \\ + \left\| \Delta\mathcal{U}(k) \right\|_{\underline{\Lambda}}^2$$

where the matrix $\mathbf{Z}^P$ is given by (3.78). In cases when disturbance prediction is not applied within the prediction horizon, the matrix $\mathbf{Z}^P$ will get reduced to the matrix $\mathbf{M}^{zP}$, (3.74).

## 3.3 Generalized Predictive Control (GPC) Algorithm

First commercial MPC algorithms were based on process models in the form of a discrete impulse response – *Model Algorithmic Control (MAC) algorithm* [119], and a discrete step response – *Dynamic Matrix Control algorithm*. The latter was presented in detail in the previous section. An MPC algorithm applying a model in the form of an impulse response is very similar, the impulse

and step responses of a dynamic process are very closely related. We decided to present the DMC algorithm as the first and basic one in this book, as the step response is a more natural description of the process dynamics. First of all, it is easier to be directly obtained in an on-line identification experiment – thus the popularity of commercial products based on the DMC algorithm. We shall not present the approach which uses the process model in the form of an impulse response. Its significance seems to be smaller, moreover, all formulae based on the step response can be transformed into the corresponding formulae using the impulse response, using the well-known relationship

$$s_k = \sum_{j=0}^{k} h_j$$

where $\{s_j\}$ and $\{h_j\}$ are sequences of the step response and impulse response coefficients, respectively.

The GPC (*Generalized Predictive Control*) algorithm turned out to be one of the most popular predictive control algorithms, it was proposed slightly later than algorithms based on non-parametric models of step or impulse responses. The GPC algorithm uses a process model in the form of a discrete difference equation describing the process input-output relation (equivalently, a discrete transfer function) [27], see also [17, 18]. In this section we shall present the GPC algorithm referring to general features of the MPC algorithms presented earlier, particularly when presenting the DMC algorithm in detail in the previous section.

In the formulation of the cost function of the GPC algorithm a *reference trajectory* $y^{ref}(k+p|k)$, $p = N_1, ..., N$, is often considered, in place of the set-point trajectory only. It is usually defined in the following way

$$y^{ref}(k+p|k) = \gamma y^{ref}(k+p-1|k) + (1-\gamma)y^{sp}(k+p|k), \quad p = 1, ..., N \quad (3.81)$$

with $y^{ref}(k|k) = y(k)$, and usually also $y^{sp}(k+p|k) = y^{sp}(k), p = 1, ..., N$ (constant set-point over the prediction horizon), see *e.g.*, [17, 18]. In the above formula $\gamma$, $0 \leq \gamma < 1$, is a parameter defining how quick the reference trajectory should approach the set-point trajectory, starting from the currently measured process output value $y(k)$. For $\gamma = 0$ the reference trajectory becomes the set-point trajectory, increasing $\gamma$ slows down the dynamics of the reference trajectory, makes it smoother – thus resulting in less stringent requirements for the controller output signal. The role and meaning of the reference trajectory will be discussed in more detail in Section 3.6.2, when discussing the role and tuning of parameters of predictive controllers. A reference trajectory can be used in every MPC algorithm as a certain filter of the difference between $y(k)$ and $y^{sp}(k+p|k)$, occurring in the cost function and all the following formulae exactly in place of the set-point trajectory $y^{sp}(k+p|k)$. Having the above statement in mind, we shall use further on the set-point trajectory in all formulae, consequently.

In the GPC algorithm a process model in the form of a discrete difference equation is used. In a general case of a multivariable process with $n_u$ control inputs and $n_y$ controlled outputs it is a model in the form

$$\mathbf{A}(z^{-1})y(k) = \mathbf{B}(z^{-1})u(k-1) + \mathbf{C}(z^{-1})\frac{\epsilon(k)}{\triangle} \tag{3.82}$$

where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are polynomial matrices

$$\mathbf{A}(z^{-1}) = \mathbf{1} + \mathbf{A}_1 z^{-1} + \mathbf{A}_2 z^{-2} + \cdots + \mathbf{A}_{n_A} z^{-n_A} \tag{3.83a}$$

$$\mathbf{B}(z^{-1}) = \mathbf{B}_0 + \mathbf{B}_1 z^{-1} + \mathbf{B}_2 z^{-2} + \cdots + \mathbf{B}_{n_B} z^{-n_B} \tag{3.83b}$$

$$\mathbf{C}(z^{-1}) = \mathbf{1} + \mathbf{C}_1 z^{-1} + \mathbf{C}_2 z^{-2} + \cdots + \mathbf{C}_{n_C} z^{-n_C} \tag{3.83c}$$

$z^{-1}$ denotes the operator of a unit time delay, $\epsilon(k)$ is a vector of white noises with zero mean value, while $\triangle = 1 - z^{-1}$ denotes the backward-difference operator (thus $1/\triangle$ denotes integration). If $z$ is treated as a complex variable of the $\mathcal{Z}$ transform then the model used can be considered as a discrete transfer function between the process inputs and outputs.

The model (3.82) is often described as an ARIMAX model (*Auto-Regressive Integrated Moving Average with eXogenous Input*), see *e.g.*, [103], it is called also a CARIMA model (*Controlled Auto-Regressive Integrated Moving Average*), see *e.g.*, [18]. In this model the white noise $\epsilon(k)$ undergoes integration, thus in effect the disturbances are non-stationary. If $\mathbf{C}(z^{-1}) = 1$ then the system is influenced by integrated white noises, if $\mathbf{C}(z^{-1}) \neq 1$ then the integrated noises are colored.

For $\mathbf{C}(z^{-1}) = 1$ the process description (3.82) takes the following form

$$\mathbf{A}(z^{-1})y(k) = \mathbf{B}(z^{-1})u(k-1) + \frac{\epsilon(k)}{\triangle} \tag{3.84}$$

This case is very important from a practical point of view. First of all, because identification of the polynomial $\mathbf{C}$ is usually difficult: parameter estimates do not converge or converge slowly [103]. Thus, this polynomial is usually treated in a different way: not as a part of the model, but as a filter with parameters undergoing a tuning process, to improve certain properties of the controller [18, 82, 123]. Moreover, in the case with $\mathbf{C}(z^{-1}) = 1$ it is easier to derive formulae describing output predictions and thus the explicit GPC control law in cases without constraints. That is why many authors consider the GPC controllers solely for this case. We shall also proceed along this lines, referring an interested reader to the book [18], devoted mainly to the GPC control algorithm in its multiple versions, see also [123].

### 3.3.1 GPC Algorithm for a SISO Process

Consider first a single-input single-output (SISO) process. Due to the higher complexity of the transformations needed to derive output predictions and

thus the explicit control law than in the case of the DMC controller, this will allow for easier understanding of the reasoning. The obtained results will then be extended to the multi-input multi-output case.

In the case of a SISO process its model (3.84) is reduced to the equation

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + \frac{\epsilon(k)}{\triangle} \tag{3.85}$$

where $A(z^{-1})$, $B(z^{-1})$ are polynomials of $z^{-1}$, respectively

$$A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_{n_A} z^{-n_A}$$
$$B(z^{-1}) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_{n_B} z^{-n_B}$$

In the above model we do not treat the delay $\tau$ separately, apart from a unitary delay connected with the discretization of the process model as a causal system. If there is a delay in the process, then first $\tau$ coefficients $b_j$ of the polynomial $B$, corresponding to this delay, will be zero.

With the same general formulation of the cost function, various linear MPC algorithms differ by the way the predicted trajectories of the outputs $y(k+p|k)$, $p = 1, ..., N$ are evaluated, see Fig. 3.1. In the GPC algorithm these values are evaluated on the basis of the process model (3.85), see *e.g.*, [17, 18]. For this purpose, let us consider the Bézout identity (Diophantine equation with $C(z^{-1}) = 1$)

$$E_p(z^{-1})\overline{A}(z^{-1}) + F_p(z^{-1})z^{-p} = 1 \tag{3.86}$$

where

$$\overline{A}(z^{-1}) = \triangle A(z^{-1})$$

while polynomials $E_p$ and $F_p$ are of degree $p-1$ and $n_A$, respectively. They can be obtained by dividing 1 by $\overline{A}(z^{-1})$ until the rest takes the form $F_p(z^{-1})z^{-N}$, see *e.g.*, [3]. The following formulae for coefficients of polynomials $E_p$ and $F_p$ follow directly:

$$\text{initial values:} \qquad e_{1,0} = 1, \quad f_{1,i} = -\bar{a}_{i+1}, \quad i = 0, ..., n_A$$

and recurrent formulae for $p = 2, ..., N$:

$$e_{p,i} = \begin{cases} e_{p-1,i} & \text{for } \ i = 0, ..., p-2 \\ f_{p-1,0} & \text{for } \ i = p-1 \end{cases}$$

$$f_{p,i} = \begin{cases} -f_{p-1,0}\bar{a}_{i+1} + f_{p-1,i+1} & \text{for } \ i = 0, ..., n_A - 1 \\ -f_{p-1,0}\bar{a}_{i+1} & \text{for } \ i = n_A \end{cases}$$

Multiplying both sides of the process model equation (3.85) by $E_p(z^{-1})z^p$ we obtain

$$E_p(z^{-1})\overline{A}(z^{-1})y(k+p) = E_p(z^{-1})B(z^{-1})\triangle u(k+p-1) + E_p(z^{-1})\epsilon(k+p)$$

Writing the Bézout identity (3.86) in the form

$$E_p(z^{-1})\overline{A}(z^{-1}) = 1 - F_p(z^{-1})z^{-p}$$

and inserting it into the previous equation we obtain

$$(1 - F_p(z^{-1})z^{-p})y(k+p) = E_p(z^{-1})B(z^{-1})\triangle u(k+p-1) + E_p(z^{-1})\epsilon(k+p)$$

It can be written in the form

$$y(k+p) = F_p(z^{-1})y(k) + E_p(z^{-1})B(z^{-1})\triangle u(k+p-1) + E_p(z^{-1})\epsilon(k+p)$$

Because the polynomial $E_p$ is of degree at most $p-1$, then the term $E_p(z^{-1})\epsilon(k+p)$ represents only *future values* of the disturbance noise. In the GPC controller a minimum-variance prediction is used, it is obtained when

$$y(k+p|k) = E\{y(k+p)|k\}$$

where $E\{\cdot\ |k\}$ denotes a conditional expected value, under conditions of information available at sampling instant $k$, see *e.g.*, [27, 26, 18, 3]. The term $F_p(z^{-1})y(k)$ contains only known (present and past) output values. Further, the terms defined by $E_p(z^{-1})B(z^{-1})\triangle u(k+p-1|k)$ contain only deterministic control input increments:

$$\triangle u(k+p-1|k) = \begin{cases} \triangle u(k+p-1|k), & p-1 \geq 0 \ - \text{ decision variables} \\ \triangle u(k+p-1), & p-1 < 0 \ - \text{ past values} \end{cases} \tag{3.87}$$

Taking into account these statements and the fact that the mean value of a sum is equal to a sum of mean values, we obtain

$$y(k+p|k) = F_p(z^{-1})y(k) + E_p(z^{-1})B(z^{-1})\triangle u(k+p-1|k) + \\ + E\{E_p(z^{-1})\epsilon(k+p)\}$$

Because the term $E_p(z^{-1})\epsilon(k+p)$ represents only future values of a zero mean white noise, then $E\{E_p(z^{-1})\epsilon(k+p)\} = 0$. Thus, we obtain a minimum-variance predictor in the following form

$$\begin{aligned} y(k+p|k) &= F_p(z^{-1})y(k) + E_p(z^{-1})B(z^{-1})\triangle u(k+p-1|k) \\ &= F_p(z^{-1})y(k) + G_p(z^{-1})\triangle u(k+p-1|k) \end{aligned} \tag{3.88}$$

where

$$G_p(z^{-1}) = E_p(z^{-1})B(z^{-1}) \tag{3.89}$$

Because the degree of the polynomial $G_p$ is $n_B + (p-1)$, then there are both future and past values of process input signal increments in (3.88), back up

to the sampling instant $k+p-1-n_B-(p-1)=k-n_B$, and $n_A+1$ known output values, from $y(k)$ back to $y(k-n_A)$.

Each sum $G_p(z^{-1})\triangle u(k+p-1|k)$, $p=1,...,N$ can be divided into a part dependent on present and future input increments, and a part dependent on past input increments, as follows

$$G_p(z^{-1})\triangle u(k+p-1|k)=G_p^{FG}(z)\triangle u(k|k)+G_p^{PG}(z^{-1})\triangle u(k-1) \quad (3.90)$$

where the order of each polynomial $G_p^{PG}(z^{-1})$ is no higher than $n_B$. Let us recall, for convenience, definitions of the vectors $\mathcal{Y}^{sp}(k)$, $\Delta\mathcal{Y}(k)$, $\mathcal{Y}^0(k)$ and $\Delta\mathcal{U}(k)$ and define vectors $\Delta\mathcal{U}^{PG}(k)$ and $\mathcal{Y}^{PG}(k)$, as follows

$$\mathcal{Y}^{sp}(k)=\begin{bmatrix} y^{sp}(k+N_1|k) \\ \vdots \\ y^{sp}(k+N|k) \end{bmatrix}, \quad \Delta\mathcal{Y}(k)=\begin{bmatrix} \triangle y(k+N_1|k) \\ \vdots \\ \triangle y(k+N|k) \end{bmatrix}$$

$$\mathcal{Y}^0(k)=\begin{bmatrix} y^0(k+N_1|k) \\ \vdots \\ y^0(k+N|k) \end{bmatrix}, \quad \Delta\mathcal{U}(k)=\begin{bmatrix} \triangle u(k|k) \\ \vdots \\ \triangle u(k+N_u-1|k) \end{bmatrix}$$

$$\Delta\mathcal{U}^{PG}(k)=\begin{bmatrix} \triangle u(k-1) \\ \vdots \\ \triangle u(k-n_B) \end{bmatrix}, \quad \mathcal{Y}^{PG}(k)=\begin{bmatrix} y(k) \\ \vdots \\ y(k-n_A) \end{bmatrix}$$

We can now write (3.90) in the following form

$$G_p(z^{-1})\triangle u(k+p-1|k)=\mathbf{g}_p^{FG}\Delta\mathcal{U}(k)+\mathbf{g}_p^{PG}\Delta\mathcal{U}^{PG}(k), \quad p=N_1,...,N$$

where $\mathbf{g}_p^{FG}$ and $\mathbf{g}_p^{PG}$ are (row) vectors consisting of coefficients of the polynomials $G_p^{FG}$ and $G_p^{PG}$ (complemented with zeroes if necessary). Denoting

$$\mathbf{G}^{FG}=\begin{bmatrix} \mathbf{g}_{N_1}^{FG} \\ \vdots \\ \mathbf{g}_N^{FG} \end{bmatrix}, \quad \mathbf{G}^{PG}=\begin{bmatrix} \mathbf{g}_{N_1}^{PG} \\ \vdots \\ \mathbf{g}_N^{PG} \end{bmatrix}$$

we can write the forced and free components of the output prediction in the following form

$$\Delta\mathcal{Y}(k)=\mathbf{G}^{FG}\Delta\mathcal{U}(k) \quad (3.91)$$

$$\mathcal{Y}^0(k)=\mathbf{F}\mathcal{Y}^{PG}(k)+\mathbf{G}^{PG}\Delta\mathcal{U}^{PG}(k) \quad (3.92)$$

It follows from the form of the forced component equation (3.91) that the equality

$$\mathbf{G}^{FG}=\mathbf{M}$$

must be true, where $\mathbf{M}$ is the dynamic matrix (3.33), derived in Section 3.2.2, for $N_1=\tau+1$ taking the characteristic form (3.35). Thus, it can be concluded that

$$
\begin{aligned}
G^{PG}_{\tau+1}(z^{-1}) &= (G_{\tau+1}(z^{-1}) - s_{\tau+1})z^1 \\
G^{PG}_{\tau+2}(z^{-1}) &= (G_{\tau+2}(z^{-1}) - s_{\tau+1} - s_{\tau+2}z^{-1})z^2 \\
&\vdots \\
G^{PG}_{\tau+j}(z^{-1}) &= (G_{\tau+j}(z^{-1}) - s_{\tau+1} - s_{\tau+2}z^{-1} - \cdots - s_{\tau+j}z^{-j+1})z^j
\end{aligned}
$$

for $j = 1, 2..., N-\tau$, where $s_{\tau+1}$ is the first non-zero coefficient of the process step response, and $s_{\tau+2}, s_{\tau+3}, ...$ are the following coefficients. Rows of the matrix $\mathbf{F}$ consist of coefficients of polynomials $F_p$,

$$
F_p = f_{p,0} + f_{p,1}z^{-1} + \cdots + f_{p,n_A}z^{-n_A}, \quad p = N_1, ..., N
$$

$$
\mathbf{F} = \begin{bmatrix}
f_{N_1,0} & f_{N_1,1} & \cdots & f_{N_1,n_A} \\
f_{N_1+1,0} & f_{N_1+1,1} & \cdots & f_{N_1+1,n_A} \\
\vdots & \vdots & \ddots & \vdots \\
f_{N,0} & f_{N,1} & \cdots & f_{N,n_A}
\end{bmatrix}
$$

Using the derived formulae, the cost function of the controller optimization problem can be written in the form (3.40)

$$
J(k) = \left\| [\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] - \mathbf{M}\Delta\mathcal{U}(k) \right\|^2_{\underline{\Psi}} + \|\Delta\mathcal{U}(k)\|^2_{\underline{\Lambda}} \tag{3.93}
$$

identical as the one formulated for the DMC algorithm in Section 3.2.2 – only the free component $\mathcal{Y}^0(k)$ will now be in the form (3.92), in place of (3.28). Reasoning which leads to an explicit, analytical from of the GPC control law (with inequality constraints not considered) will thus be the same as it was in Section 3.2.2 for the DMC controller. In this way we obtain the vector of optimal increments of the controller output signal in the form

$$
\Delta\widehat{\mathcal{U}}(k) = \mathbf{K}[\mathcal{Y}^{sp}(k) - (\mathbf{F}\mathcal{Y}^{PG}(k) + \mathbf{G}^{PG}\Delta\mathcal{U}^{PG}(k))] \tag{3.94}
$$

and the *explicit, unconstrained GPC control law*

$$
\triangle\hat{u}(k) = \triangle\hat{u}(k|k) = \overline{\mathbf{K}}_1[\mathcal{Y}^{sp}(k) - (\mathbf{F}\mathcal{Y}^{PG}(k) + \mathbf{G}^{PG}\Delta\mathcal{U}^{PG}(k))] \tag{3.95}
$$

in a typical form of a linear dependence relating the optimal control increments to the difference between a set-point trajectory and a free output trajectory, where $\overline{\mathbf{K}}_1$ denotes the first row of the matrix $\mathbf{K}$, see (3.44), (3.46), (3.65).

We shall present now the GPC control law in a form showing a direct dependence of the current optimal controller output $\triangle\hat{u}(k)$ on past process input increments (*i.e.*, past controller outputs) and on process outputs. Denoting

$$
\mathbf{F} = \begin{bmatrix} \mathbf{F}_0 & \mathbf{F}_1 & \cdots & \mathbf{F}_{n_A} \end{bmatrix}
$$

$$
\mathbf{G}^{PG} = \begin{bmatrix} \mathbf{G}^{PG}_1 & \mathbf{G}^{PG}_2 & \cdots & \mathbf{G}^{PG}_{n_B} \end{bmatrix}
$$

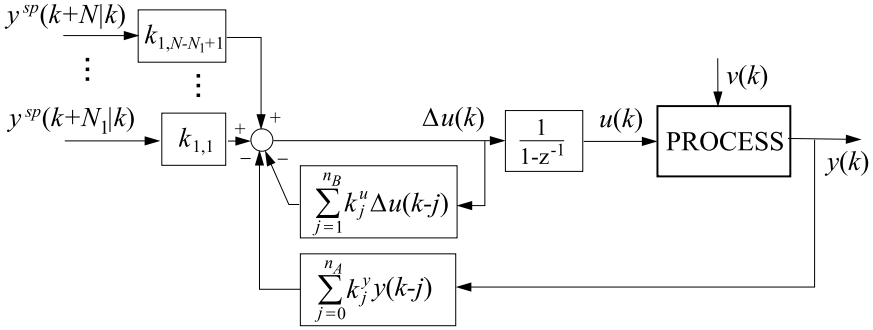


**Fig. 3.20.** Structure of the explicit unconstrained GPC control law

where $\mathbf{F}_j$ and $\mathbf{G}_j^{PG}$ are columns of the matrices $\mathbf{F}$ and $\mathbf{G}^{PG}$, and using the structure (3.46) of the matrix $\mathbf{K}$ we can write

$$\begin{aligned}\triangle\hat{u}(k) &= \sum_{p=N_1}^{N} k_{1,p-N_1+1} y^{sp}(k+p|k) + \\ &\quad - \sum_{j=0}^{n_A} (\overline{\mathbf{K}}_1 \mathbf{F}_j) y(k-j) - \sum_{j=1}^{n_B} (\overline{\mathbf{K}}_1 \mathbf{G}_j^{PG}) \triangle u(k-j) \\ &= \sum_{p=N_1}^{N} k_{1,p-N_1+1} y^{sp}(k+p|k) + \\ &\quad - \sum_{j=0}^{n_A} k_j^y y(k-j) - \sum_{j=1}^{n_B} k_j^u \triangle u(k-j) \end{aligned} \tag{3.96}$$

where, due to the considered case of a SISO system, the appropriate controller gains $k_{1p}$, $k_j^y$ and $k_j^u$ are scalars. The obtained structure of the *unconstrained, explicit version of the GPC control law* is presented in Fig. 3.20.

In a practically important case with the set-point constant in the entire prediction horizon, *i.e.*, $y^{sp}(k+1|k) = \cdots = y^{sp}(k+N|k) = y^{sp}(k)$, the GPC control law (3.96) reduces to the form

$$\triangle\hat{u}(k) = k^e y^{sp}(k) - \sum_{j=0}^{n_A} k_j^y y(k-j) - \sum_{j=1}^{n_B} k_j^u \triangle u(k-j) \tag{3.97}$$

where

$$k^e = \sum_{p=N_1}^{N} k_{1,p-N_1+1}$$

The structure of the GPC control law (3.96), or (3.97), corresponds to the one met in the literature, see *e.g.*, [18]. However, it is not very convenient for certain comparisons and analyses. Therefore, it will now be transformed to a

slightly different form. To this end, a lemma presenting the detailed structure of the matrix **F** must be proven first. Recall that rows of the matrix **F** consist of coefficients of polynomials $F_p(z^{-1})$, $p = N_1, ..., N$, obtained from solutions of the Bézout identities (3.86).

**Lemma 3.1** *Each polynomial*

$$F_p(z^{-1}) - 1 = f_{p,0} - 1 + f_{p,1}z^{-1} + \cdots + f_{p,n_A}z^{-n_A}, \quad p = 1, 2, ..., N$$

*can be divided by* $\triangle = 1 - z^{-1}$.

*Proof.* The thesis will be proven using the principle of mathematical induction. For $p = 1$ the Bézout identity (3.86) takes the form

$$E_1(z^{-1})\overline{A}(z^{-1}) + F_1(z^{-1})z^{-1} = 1$$

Subtracting $z^{-1}$ from both sides we obtain

$$E_1(z^{-1})\triangle A(z^{-1}) + (F_1(z^{-1}) - 1)z^{-1} = \triangle$$

because, from the definition, $\overline{A}(z^{-1}) = \triangle A(z^{-1})$. Thus, if the polynomial $F_1(z^{-1})$ is a solution to the Bézout identity, then the polynomial $F_1(z^{-1}) - 1$ must be divisible by $\triangle$.

Assume now that for a certain $p \geq 1$ the polynomial $F_p(z^{-1}) - 1$ is divisible by $\triangle$. The following recurrent formula is a direct consequence of the fact that the polynomials $E_p$ and $F_p$ were obtained as a result of the division of 1 by the polynomial $\overline{A}(z^{-1})$:

$$F_{p+1}(z^{-1})z^{-1} = F_p(z^{-1}) - f_{p,0}\overline{A}(z^{-1})$$

Subtracting $z^{-1}$ from the left side of this equality and $z^{-1} = z^{-1} - 1 + 1$ from its right side we obtain

$$(F_{p+1}(z^{-1}) - 1)z^{-1} = (F_p(z^{-1}) - 1) + (1 - z^{-1}) - f_{p,0}\overline{A}(z^{-1})$$

Because $\overline{A}(z^{-1}) = \triangle A(z^{-1})$, then if $F_p(z^{-1}) - 1$ is divisible by $\triangle$, then $F_{p+1}(z^{-1}) - 1$ must be divisible by $\triangle$, too. This completes the proof. □

Using the lemma, each polynomial $F_p(z^{-1})$ can be presented in the following form

$$\begin{aligned} F_p(z^{-1}) &= 1 + \bar{F}_p(z^{-1})\triangle \\ &= 1 + \bar{f}_{p,1}\triangle + \bar{f}_{p,2}z^{-1}\triangle + \cdots + \bar{f}_{p,n_A}z^{-n_A+1}\triangle \end{aligned} \tag{3.98}$$

where

$$\bar{f}_{p,1} = f_{p,0} - 1$$

$$\bar{f}_{p,,j+1} = f_{p,j} + \bar{f}_{p,j}, \quad j = 2, ..., n_A - 1$$

Therefore, the following alternative representation of the product $\mathbf{F}\,\mathcal{Y}^{PG}(k)$ follows directly

$$\mathbf{F}\,\mathcal{Y}^{PG}(k) = \mathbf{I}\,y(k) + \begin{bmatrix} \bar{f}_{N_1,1} & \bar{f}_{N_1,2} & \cdots & \bar{f}_{N_1,n_A} \\ \bar{f}_{N_1+1,1} & \bar{f}_{N_1+1,2} & \cdots & \bar{f}_{N_1+1,n_A} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{f}_{N,1} & \bar{f}_{N,2} & \cdots & \bar{f}_{N,n_A} \end{bmatrix} \begin{bmatrix} \triangle y(k) \\ \triangle y(k-1) \\ \vdots \\ \triangle y(k-n_A+1) \end{bmatrix}$$

$$= \mathbf{I}\,y(k) + \bar{\mathbf{F}}\,\triangle\mathcal{Y}^{PG}(k) \tag{3.99}$$

where $\triangle\mathcal{Y}^{PG}(k) = [\triangle y(k)\ \triangle y(k-1)\ \cdots\ \triangle y(k-n_A+1)]^T$. Denoting columns of the matrix $\bar{\mathbf{F}}$ by $\bar{\mathbf{F}}_j$,

$$\bar{\mathbf{F}} = [\bar{\mathbf{F}}_1\ \cdots\ \bar{\mathbf{F}}_{n_A}]$$

the GPC control law (3.96) can now be transformed to the form

$$\begin{aligned} \triangle\hat{u}(k) &= \sum_{p=N_1}^{N} k_{1,p-N_1+1}(y^{sp}(k+p|k) - y(k)) + \\ &\quad - \sum_{j=1}^{n_A}(\overline{\mathbf{K}}_1\bar{\mathbf{F}}_j)\triangle y(k-j+1) - \sum_{j=1}^{n_B} k_j^u \triangle u(k-j) \\ &= \sum_{p=N_1}^{N} k_{1,p-N_1+1}(y^{sp}(k+p|k) - y(k)) + \\ &\quad - \sum_{j=1}^{n_A} \bar{k}_j^y \triangle y(k-j+1) - \sum_{j=1}^{n_B} k_j^u \triangle u(k-j) \end{aligned} \tag{3.100}$$

which in the case of the set-points constant in the entire prediction horizon reduces to

$$\triangle\hat{u}(k) = k^e(y_k^{sp} - y(k)) - \sum_{j=1}^{n_A} \bar{k}_j^y \triangle y(k-j+1) - \sum_{j=1}^{n_B} k_j^u \triangle u(k-j) \tag{3.101}$$

Figures 3.21 and 3.22 illustrate the obtained alternative structure of the GPC control law.

In cases with constraints on the process input signal, it is necessary to complement the structure of the GPC explicit unconstrained controller, analogously as it was done for the DMC controller – adding to the controller structure nonlinear elements implementing constraints on controller output increments and amplitudes and adding the anti-windup loop. An example of such a structure, being an extension of Fig. 3.22, is presented in Fig. 3.23 (compare with Fig. 3.14).

The GPC control law presented in the form (3.100), or (3.101) for a constant set-points prediction, is easy to *compare with the DMC control law* given by the formula (3.48), or (3.52), respectively. Analogously, we can compare

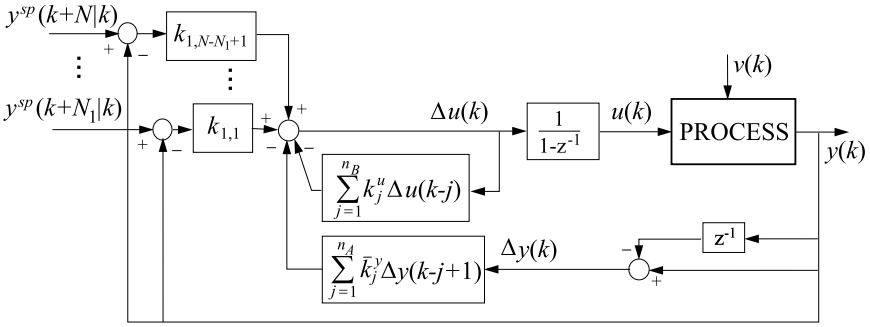


**Fig. 3.21.** Alternative structure of the unconstrained GPC control law

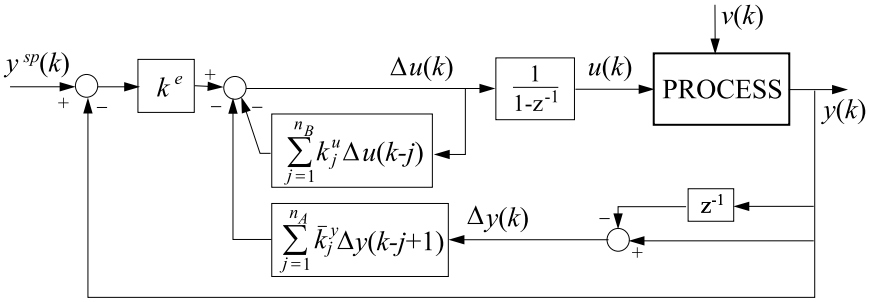


**Fig. 3.22.** Alternative structure of the unconstrained GPC control law for the set-point constant in the prediction horizon

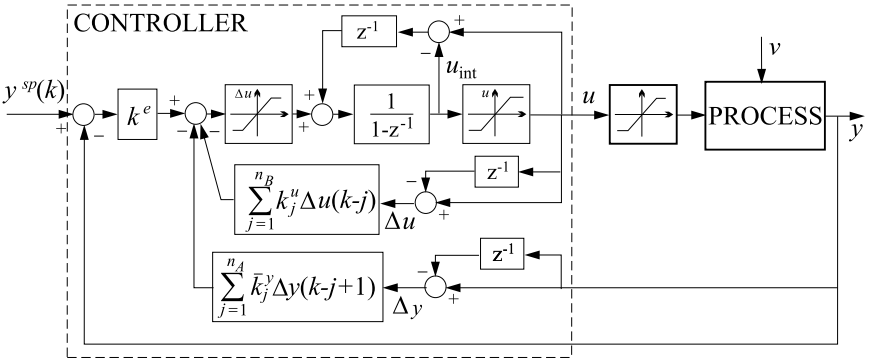


**Fig. 3.23.** Structure of the unconstrained GPC control law from the previous figure, supplemented by elements performing projection of its output onto constraints on amplitude and rate of change of process inputs, with additional anti-windup loop

Figures 3.21 and 3.22 illustrating the structure of the explicit GPC controller with Figures 3.4 and 3.5 which present the structure of the explicit DMC controller. In the DMC controller structure, there is one feedback from the process output (only current output value $y(k)$ is fed back) and feedbacks from $D-1$ values of past process input increments (where $D$ denotes a dynamics horizon of the process). In the GPC controller, there are feedbacks from the current process output value and last $n_A$ past values of the output increments, and from last $n_B$ past values of the process input increments, where $n_A$ and $n_B$ are degrees of polynomials of the process model (3.85).

If the sampling period is short in comparison with the dynamics horizon of the process, then the step response contains a large number of elements, it is "long". At the same time, it can be often well described by a discrete transfer function with polynomials of a relatively low order. Then $D \gg n_A$ and $D \gg n_B$ and a model used in the GPC algorithm has a much more efficient representation, the number of feedbacks in the structure of the control law is much smaller. However, remember that a model in the form of a step response is non-parametric, obtained most often on the basis of a direct, simple and comprehensive identification experiment on the process. To obtain a model used in the GPC algorithm it is necessary to perform an appropriate identification procedure both of its structure and parameters.

**Example 3.2**
To illustrate the design method and to present the form of the GPC control law, it will be calculated for the process from Example 3.1, modeled there by a discrete step response (3.55). Assume, as in the mentioned example, that $N_1 = 3$, $N_u = 3$, $N = 6$ and $\underline{\mathbf{\Psi}} = \mathbf{I}$, $\underline{\mathbf{\Lambda}} = \lambda\mathbf{I}$. For the GPC control algorithm, the process will be described by a model in the form of a discrete difference equation

$$y(k) + ay(k-1) = b_2 u(k-3) + b_3 u(k-4) + \epsilon(k)/\triangle$$

which describes dynamics with the process time delay $\tau$ equal to two sampling periods (see the process step response in Example 3.1), and thus the corresponding input time delay in the difference equation $\bar{\tau} = 3$. Fitting, for $\epsilon(k) = 0$, the model parameters to the points of the step response (by the least squares method) we obtain

$$(1 - 0.2676z^{-1})y(k) = (0.1989z^{-2} + 0.2552z^{-3})u(k-1) \tag{3.102}$$

*i.e.*,

$$A(z^{-1}) = 1 - 0.2676z^{-1}$$
$$B(z^{-1}) = 0.1989z^{-2} + 0.2552z^{-3}$$

The Bézout identity in the considered case takes the following form

$$E_p(z^{-1})(1 - 1.2676z^{-1} + 0.2676z^{-2}) + F_p(z^{-1})z^{-p} = 1$$

where

$$1 - 1.2676z^{-1} + 0.2676z^{-2} = A(z^{-1})(1 - z^{-1}) = \overline{A}(z^{-1})$$

The solutions to the Bézout identity for subsequent values of $p = 1, 2, ..., 6$, obtained by a division of 1 by $\overline{A}(z^{-1})$, are presented in Table 3.1. In the last column of the table polynomials $\bar{F}_p(z^{-1})$ which are transformed forms (3.98) of the polynomials $F_p(z^{-1})$, are also given.

**Table 3.1.** Polynomial solutions to the Bézout identity

| p | $E_p(z^{-1})$ | $F_p(z^{-1})$ | $\bar{F}_p(z^{-1})$ |
|---|---|---|---|
| 1 | $1$ | $1.2676 - 0.2676z^{-1}$ | 0.2676 |
| 2 | $1 + 1.2676z^{-1}$ | $1.3392 - 0.3392z^{-1}$ | 0.3392 |
| 3 | $1 + 1.2676z^{-1} + 1.3392z^{-2}$ | $1.3584 - 0.3584z^{-1}$ | 0.3584 |
| 4 | $1 + 1.2676z^{-1} + 1.3392z^{-2} + +1.3584z^{-3}$ | $1.3635 - 0.3635z^{-1}$ | 0.3635 |
| 5 | $1 + 1.2676z^{-1} + 1.3392z^{-2} + +1.3584z^{-3} + 1.3635z^{-4}$ | $1.3649 - 0.3649z^{-1}$ | 0.3649 |
| 6 | $1 + 1.2676z^{-1} + 1.3392z^{-2} + +1.3584z^{-3} + 1.3635z^{-4} + +1.3649z^{-5}$ | $1.3652 - 0.3652z^{-1}$ | 0.3652 |

Next, calculating polynomials $G_p(z^{-1}) = E_p(z^{-1})B(z^{-1})$, the searched formulae (3.88) for predictions $y(k+p|k)$, $p = 3, 4, 5, 6$ are obtained. We write these formulae in a vector form (3.91) and (3.92) below, separately for the forced component $\Delta\mathcal{Y}(k)$ and for the free component $\mathcal{Y}^0(k)$

$$\begin{bmatrix} \triangle y(k+3|k) \\ \triangle y(k+4|k) \\ \triangle y(k+5|k) \\ \triangle y(k+6|k) \end{bmatrix} = \begin{bmatrix} 0.1989 & 0 & 0 \\ 0.5073 & 0.1989 & 0 \\ 0.5899 & 0.5073 & 0.1989 \\ 0.6120 & 0.5899 & 0.5073 \end{bmatrix} \begin{bmatrix} \triangle u(k|k) \\ \triangle u(k+1|k) \\ \triangle u(k+2|k) \end{bmatrix}$$

$$i.e., \quad \Delta\mathcal{Y}(k) = \mathbf{M}\,\Delta\mathcal{U}(k)$$

$$\begin{bmatrix} y^0(k+3|k) \\ y^0(k+4|k) \\ y^0(k+5|k) \\ y^0(k+6|k) \end{bmatrix} = \begin{bmatrix} 0.5073 & 0.5899 & 0.3418 \\ 0.5899 & 0.6120 & 0.3467 \\ 0.6120 & 0.6183 & 0.3485 \\ 0.6183 & 0.62 & 0.3483 \end{bmatrix} \begin{bmatrix} \triangle u(k-1) \\ \triangle u(k-2) \\ \triangle u(k-3) \end{bmatrix} + + \begin{bmatrix} 1.3584 & -0.3584 \\ 1.3635 & -0.3635 \\ 1.3649 & -0.3649 \\ 1.3652 & -0.3652 \end{bmatrix} \begin{bmatrix} y(k) \\ y(k-1) \end{bmatrix}$$

$$i.e., \quad \mathcal{Y}^0(k) = \mathbf{G}^{PG}\Delta\mathcal{U}^{PG}(k) + \mathbf{F}\,\mathcal{Y}^{PG}(k)$$

A careful reader should note certain differences in values of elements of the dynamic matrices $\mathbf{M}$, the one obtained above and that from Example 3.1 (compare with (3.56)). The differences are caused by the fact that the matrix obtained above consists of elements of the step response of the dynamic system described by (3.102), and not of the elements of the original step response (3.55).

A solution to the GPC optimization problem is given by the formula (3.94), where matrices $\mathbf{K}$ are, for the same values of $\lambda$, very close to those obtained in Example 3.1, with differences resulting from slightly different matrices $\mathbf{M}$. Assuming that $y^{sp}(k+3|k) = \cdots = y^{sp}(k+6|k) = y^{sp}(k)$ we obtain the control law in the form (3.96)

$$\begin{aligned}\triangle\hat{u}(k) &= \sum_{p=1}^{4} k_{1p} y^{sp}(k) - \sum_{j=0}^{1} (\overline{\mathbf{K}}_1 \mathbf{F}_{j+1}) y(k-j) - \sum_{j=1}^{3} (\overline{\mathbf{K}}_1 \mathbf{G}_j^{PG}) \triangle u(k-j) \\ &= k^e y^{sp}(k) - \sum_{j=0}^{1} k_j^y y(k-j) - \sum_{j=1}^{3} k_j^u \triangle u(k-j)\end{aligned}$$

where $\overline{\mathbf{K}}_1$ is the first row of the matrix $\mathbf{K}$, while $\mathbf{F}_j$ and $\mathbf{G}_j^{PG}$ are columns of the matrices $\mathbf{F}$ and $\mathbf{G}^{PG}$.
For $\lambda = 0.01$ we obtain the following parameters of the GPC control law:

$$\begin{aligned}&k^e = 2.8618, \quad [k_0^y \; k_1^y] = [3.8933 \; -1.0314], \\ &[k_1^u \; k_2^u \; k_3^u] = [1.5454 \; 1.7131 \; 0.9835],\end{aligned}$$

for $\lambda = 0.05$ these parameters are

$$\begin{aligned}&k^e = 2.0653, \quad [k_0^y \; k_1^y] = [2.8177 \; -0.7464], \\ &[k_1^u \; k_2^u \; k_3^u] = [1.1484 \; 1.2453 \; 0.7120],\end{aligned}$$

and for $\lambda = 0.1$ they are equal to

$$\begin{aligned}&k^e = 1.7153, \quad [k_0^y \; k_1^y] = [2.3362 \; -0.6208], \\ &[k_1^u \; k_2^u \; k_3^u] = [0.9678 \; 1.0381 \; 0.5922].\end{aligned}$$

Let us note that absolute values of feedback coefficients $k^e$, $k_j^u$ and $k_j^y$ decrease along with the increase of the value of the weight (penalty) coefficient $\lambda$, as it was in the case of the DMC algorithm.

The control system with the designed GPC algorithm was first simulated without constraints. We shall not present the obtained trajectories, as they are practically identical to those presented in Example 3.1, obtained with the DMC algorithm, see Fig. 3.6, Fig. 3.7 and Fig. 3.8. Similar trajectories to those obtained with the DMC algorithm were also obtained for the case with constraints on amplitude and rate of change of the process input signal, applying the structure presented in Fig. 3.23, see Fig. 3.13 and Fig. 3.16, as well as for a change of the disturbance, see Fig. 3.17 and Fig. 3.18.

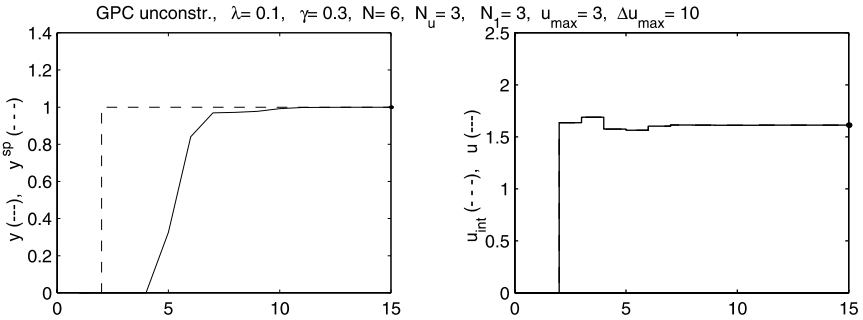


**Fig. 3.24.** Trajectories in a control system with the unconstrained GPC controller with the reference trajectory, for $\gamma = 0.3$

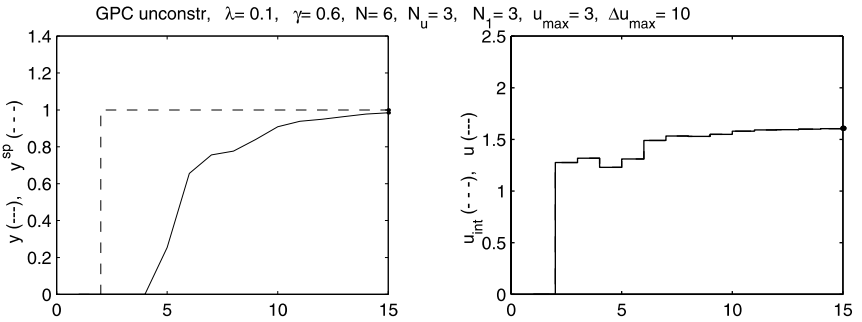


**Fig. 3.25.** Trajectories in a control system with the unconstrained GPC controller with the reference trajectory, for $\gamma = 0.6$

Trajectories presented in Fig. 3.24 and Fig. 3.25 were obtained using a reference trajectory (3.81), instead of the constant set-point trajectory, in the the GPC algorithm with $\lambda = 0.1$. They illustrate the influence of the reference trajectory, with $\gamma = 0.3$ and $\gamma = 0.6$, respectively, on the behavior of the GPC control system (compare also with Fig. 3.6 corresponding to $\gamma = 0$).

Application of the reference trajectory causes smoother changes of the controlled variable, the greater the value of $\gamma$ the longer the time needed to approach the constant set-point by the controlled variable. This result could of course be achieved only as a result of an appropriate shaping of the process input trajectory, by smoothing its variability. However, the influence of the reference trajectory on the controller output signal seems to be weaker than that caused by the weighting coefficient $\lambda$. The nature of the reference trajectory will be discussed in more detail at the end of this chapter. □

Comparing Examples 3.2 and 3.1, concerning the same process but with the GPC and DMC controllers, it should be emphasized that the considered

step response of the process (3.55) is very short, it contains only 6 elements ($D = 6$). Thus the number of coefficients of this step response and the number of parameters of the corresponding discrete transfer function model are similar. This is a result of a large sampling period, in comparison to the process dynamics. This is usually not the case – in typical situations the number of elements of a step response model is larger than the number of coefficients of a discrete transfer function model (a difference equation model).

The fact that practically identical trajectories were obtained in simulations of both GPC and DMC control systems does need to be commented upon. The output predictor in the GPC algorithm has been derived as a minimum variance one, for the model (3.85) with the disturbance in the form of an integrated zero-mean white noise $\epsilon(k)$. However, this can be equivalently written in the form

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + d(k) \tag{3.103}$$

where $d(k)$ is governed by the difference equation

$$d(k) = d(k-1) + \epsilon(k) \tag{3.104}$$

and is precisely the disturbance model used in the DMC algorithm, as optimal prediction of $d(k)$, as defined by (3.104), is the constant output disturbance prediction. Therefore, the GPC algorithm can also be derived using directly the model (3.103), as it will be shown in the next section, other formulae for prediction than those in the DMC algorithm stemming from a different linear modeling of the same process. Thus, with the same form of the cost function identical simulation results must have been obtained in Examples 3.1 and 3.2, for the same simulation conditions with exact process models assumed.

However, in cases when during a control system simulation there is a difference between a process description and its model used for the controller design, the results obtained with DMC and GPC controllers can differ. A linear model is usually only an approximate description of a nonlinear process at an operating point. Therefore, if disturbances or changes of set-points lead the process out of the range of a good linear approximation, then the effects of inaccurate modeling become visible, usually in different ways depending on the type of the linear controller applied. To check this effect, the control system was simulated with the unconstrained GPC controller from Example 3.2 in the structure where the controller output signal is not projected on process input constraints, under conditions identical to those assumed for the unconstrained DMC controller presented in Figures 3.9 and 3.15. In the first case, when an amplitude constraint is active, trajectories obtained with the GPC algorithm are presented in Fig. 3.26. First parts of the trajectories are quite similar to those presented in Fig. 3.9, but then the differences become significant. Further, in Fig. 3.27 trajectories for a case with additional and active constraint on the rate of change of the process input signal are presented. Comparing with the trajectories from Fig. 3.15, corresponding to the same situation but with the DMC controller, significant differences can be observed.

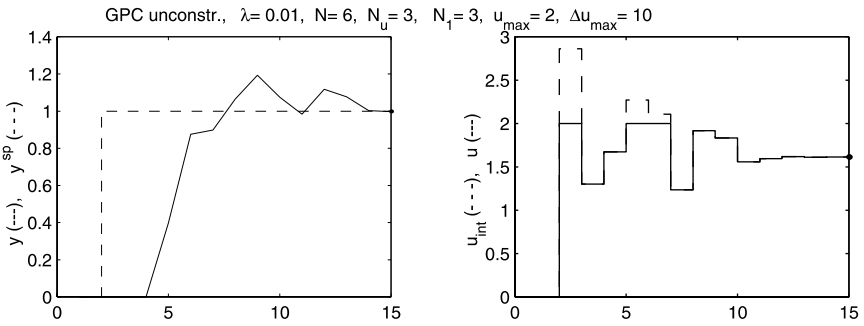


**Fig. 3.26.** Trajectories in the control system with the unconstrained GPC controller, with the controller output not projected onto the active amplitude constraint

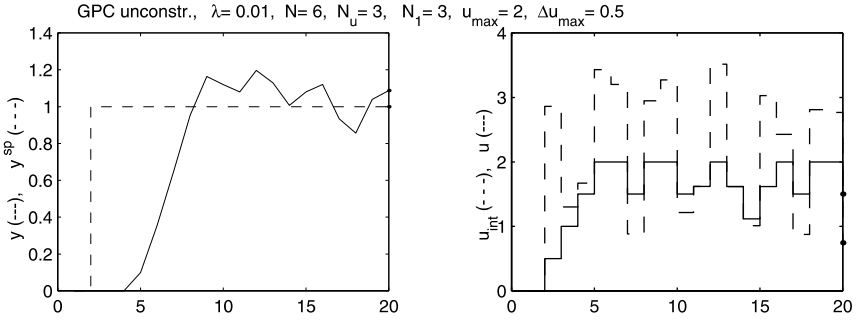


**Fig. 3.27.** Trajectories in the control system with the unconstrained GPC controller, with the controller output not projected onto active amplitude ($\leq 2$) and rate of change ($\leq 0.5$) constraints

The GPC algorithm, like the DMC algorithm, ensures zero steady-state control errors. This fact can easily be shown, reasoning analogously as we did for the DMC algorithm in the first part of Section 3.2.5. The prediction equations used should only be in the alternative form

$$y(k+p|k) = y(k) + \sum_{j=1}^{n_A} \bar{f}_{p,j} \triangle y(k-j+1) + \sum_{j=1}^{n_B} g_{p,j}^{PG} \triangle u(k-j) + \\ + \sum_{j=1}^{t} g_{p,j}^{FG} \triangle u(k+t-j|k), \quad t = \min\{p,\ N_u\}, \quad p = N_1, ..., N$$

which in the vector notation takes the following form

$$\mathcal{Y}^{pred}(k) = \mathcal{Y}(k) + \bar{\mathbf{F}} \Delta \mathcal{Y}^{PG}(k) + \mathbf{G}^{PG} \Delta \mathcal{U}^{PG}(k) + \mathbf{M} \Delta \mathcal{U}(k)$$

Now, it should only be noted that in a steady-state all increments of process inputs and outputs are zero.

A more significant difference between the GPC and DMC algorithms results from the fact that a step response model can only describe *asymptotically stable processes* (self-regulating processes), with integration at most, when incremental step response model must be used, see Section 3.2.1. On the other hand, *a transfer function model can also describe unstable processes*, with unstable poles of a discrete transfer function. To ensure a correct operation of the GPC controller in such cases, it is important to make an appropriate selection of control and prediction horizons. They have to be sufficiently long, such as to suppress influence of unstable process modes within a time interval which does not exceed the length of the prediction horizon, see [125, 82].

### 3.3.2 GPC with Constant Output Disturbance Prediction

The presented design method of the GPC algorithm, based on application of a diophantine equation (Bézout identity), is general and allows for a full formulation and analysis – but it is fairly complicated. The formulae defining the GPC algorithm can be derived in a simpler way, when assuming the process and disturbance models in the form (3.103)-(3.104), with the resulting constant output disturbance prediction, as in the DMC algorithm. This design method will now be presented.

Assuming a unitary step change in the input signal and using an ARX-type process model

$$y(k) = -a_1 y(k-1) \ \cdots - a_{n_A} y(k-n_A) + b_0 u(k-1) + \cdots + b_{n_B} u(k-n_B-1) \tag{3.105}$$

it is possible to evaluate directly the sequence of the step response coefficients $\{s_1, s_2, \ldots\}$. They are then given by the following formula:

$$s_k = - \sum_{i=1}^{\min\{k-1, n_A\}} a_i s_{k-i} + \sum_{i=0}^{\min\{k-1, n_B\}} b_i \tag{3.106}$$

The dynamic matrix $\mathbf{M}$ consists of the elements of the step response, it is used to calculate the forced component of the predicted output trajectory (3.91).

To calculate elements $y^0(k+p|k)$ of the free component of the predicted output trajectory, it is assumed that the process input is constant over the prediction horizon and equal to the value $u(k-1)$, calculated and applied in the process during the last sampling period,

$$u(k+p|k) = u(k-1), \quad p = 0, 1, ..., N-1 \tag{3.107}$$

Assuming these input signal values we shall calculate $y^0(k+p|k)$ sequentially for $p = 1, ..., N$, from the formula

$$y^0(k+p|k) = y(k+p) + d(k) \tag{3.108}$$

where subsequent values $y(k+p)$ will be calculated from the model (3.105) applied for $k = k+p$, but for $k+p-j > k$ taking, in place of unknown future values $y(k+p-j)$, their previously calculated predictions $y^0(k+p-j|k)$. To the outputs predicted in this way a constant output disturbance estimate is added, with a value $d(k)$ equal to that calculated at the current sampling instant $k$,

$$\begin{aligned} d(k) &= y(k) - y(k|k-1) \\ &= y(k) - [-\sum_{i=1}^{n_A} a_i y(k-i) + \sum_{i=0}^{n_B} b_i u(k-1-i)] \end{aligned} \tag{3.109}$$

The presented method leads to the following formula for elements of the free output response

$$\begin{aligned} y^0(k+p|k) = &- \sum_{i=1}^{\min\{n_A,p-1\}} a_i y^0(k+p-i|k) - \sum_{i=\min\{n_A,p-1\}+1}^{n_A} a_i y(k+p-i) \\ &+ \sum_{i=0}^{\min\{n_B,p\}} b_i u(k-1) + \sum_{i=\min\{n_B,p\}+1}^{n_B} b_i u(k-1+p-i) + d(k) \end{aligned} \tag{3.110}$$

$p = 1, 2, ..., N$, where the disturbance value $d(k)$ is given by (3.109).

Note that the formula (3.110) is recurrent. It does not give a general, explicit form of the dependence of the free output response trajectory on past process inputs and outputs, thus it cannot be directly used to formulate the control law. It can, however, be used during implementation of a numerical version of the GPC algorithm, where at each step the free output response is first calculated and then the quadratic optimization problem is solved. It is also possible to evaluate an explicit dependence of the free output response trajectory on past inputs and outputs only, by a recurrent application of the formula (3.110). This will be demonstrated in an example below.

**Example 3.3**

For the process from Example 3.2, we shall show how to derive formulae for the free output trajectory using (3.110).

The process model is given by (3.102), *i.e.*,

$$\begin{aligned} &a_1 = -0.2676, \;\; n_A = 1 \\ &b_0 = b_1 = 0, \; b_2 = 0.1989, \; b_3 = 0.2552, \;\; n_B = 3 \end{aligned}$$

From (3.109) we get

$$d(k) = y(k) + a_1 y(k-1) - b_2 u(k-3) - b_3 u(k-4)$$

For $p = 1$, we obtain from (3.110)

$$\begin{aligned} y^0(k+1|k) &= -a_1 y(k) + b_2 u(k-2) + b_3 u(k-3) + \\ &\quad + y(k) + a_1 y(k-1) - b_2 u(k-3) - b_3 u(k-4) \\ &= 1.2676 y(k) - 0.2676 y(k-1) + \\ &\quad + 0.1989 u(k-2) + 0.0563 u(k-3) - 0.2552 u(k-4) \end{aligned}$$

Further, for $p = 2$ we have from (3.110)

$$\begin{aligned} y^0(k+2|k) &= -a_1 y^0(k+1|k) + b_2 u(k-1) + b_3 u(k-2) + \\ &\quad + y(k) + a_1 y(k-1) - b_2 u(k-3) - b_3 u(k-4) \end{aligned}$$

Inserting the formula for $y^0(k+1|k)$ obtained earlier into this equality we obtain

$$\begin{aligned} y^0(k+2|k) &= 1.3392 y(k) - 0.3392 y(k-1) + \\ &\quad + 0.1989 u(k-1) + 0.3084 u(k-2) - 0.1838 u(k-3) - 0.3235 u(k-4) \end{aligned}$$

Analogously, for $p = 3$ we have

$$\begin{aligned} y^0(k+3|k) &= -a_1 y^0(k+2|k) + b_2 u(k-1) + b_3 u(k-1) + \\ &\quad + y(k) + a_1 y(k-1) - b_2 u(k-3) - b_3 u(k-4) \end{aligned}$$

and inserting the formula for $y^0(k+2|k)$ obtained earlier into this equality we obtain

$$\begin{aligned} y^0(k+3|k) &= 1.3584 y(k) - 0.3584 y(k-1) + \\ &\quad + 0.5073 u(k-1) + 0.0825 u(k-2) - 0.2481 u(k-3) + 0.3418 u(k-4) \end{aligned}$$

As can easily be compared, the obtained expression is equivalent to the formula calculated in Example 3.2, given by the first row of the matrix dependence describing $\mathcal{Y}^0(k)$ obtained there. Derivation of the remaining formulae, describing $y^0(k+4|k)$, $y^0(k+5|k)$ and $y^0(k+6|k)$, can be done in an analogous way, the calculations are left to the reader. □

### 3.3.3 GPC Algorithm for a MIMO Process

In the case of a multi-input multi-output (MIMO) process, with $n_u$ control inputs and $n_y$ controlled outputs, the process model is given by (3.84)

$$\mathbf{A}(z^{-1}) y(k) = \mathbf{B}(z^{-1}) u(k-1) + \frac{\epsilon(k)}{\triangle} \tag{3.111}$$

The analysis based on application of the Bézout identity and leading to the derivation of formulae for output prediction and GPC control laws, performed for a SISO process, applies directly also to the case of a MIMO process. The Bézout identity now has a matrix form (compare with (3.86))

$$\mathbf{E}_p(z^{-1})\overline{\mathbf{A}}(z^{-1}) + \mathbf{F}_p(z^{-1})z^{-p} = \mathbf{1}$$

where $\overline{\mathbf{A}}(z^{-1}) = \triangle\mathbf{A}(z^{-1})$. Reasoning analogously as it was done in the SISO case we obtain the following formula for output predictions

$$\begin{aligned} y(k+p|k) &= \mathbf{F}_p(z^{-1})y(k) + \mathbf{E}_p(z^{-1})\mathbf{B}(z^{-1})\triangle u(k+p-1|k) \\ &= \mathbf{F}_p(z^{-1})y(k) + \mathbf{G}_p(z^{-1})\triangle u(k+p-1|k) \end{aligned}$$

where $\mathbf{G}_p(z^{-1})\triangle u(k+p-1|k)$ can be divided into a term which is dependent on current and future increments of process inputs (*i.e.*, decision variables of the controller) and a term dependent on previous increments of process inputs, in this way obtaining formulae for output predictions in the form identical to (3.91) and (3.92)

$$\Delta\mathcal{Y}(k) = \mathbf{M}\,\Delta\mathcal{U}(k) \tag{3.112}$$

$$\mathcal{Y}^0(k) = \mathbf{F}\mathcal{Y}^{PG}(k) + \mathbf{G}^{PG}\Delta\mathcal{U}^{PG}(k) \tag{3.113}$$

Now, $\Delta\mathcal{Y}(k)$, $\mathcal{Y}^0(k)$ *etc.*, consist of vectors (not scalars) corresponding to the particular samples. The structure of the matrix $\mathbf{F}$ is also analogous, defined by Lemma 3.1, thus

$$\mathbf{F}\mathcal{Y}^{PG}(k) = \mathbf{I}y(k) + \bar{\mathbf{F}}\Delta\mathcal{Y}^{PG}(k)$$

The structures of explicit GPC control laws will also be the same. Figures illustrating this structure will be analogous to Figures (3.20) and (3.21) – only in all formulae scalar coefficients of controller gains should be replaced by matrices of coefficients, just like blocks of the discrete integrator and one-step delay (compare with related figures for the DMC control law in Section 3.2.2, all presented for the MIMO case there).

A detailed presentation and discussion of the GPC control algorithm for a multivariable process is beyond the scope of this book, especially because we have full analogy with the SISO case mentioned before. We refer an interested reader to the book [18], devoted mainly to the GPC algorithm. However, one will not find there the GPC control law given by the alternative formulae (3.100) or (3.101), as well as formulae for a direct, recurrent calculation of elements of a MIMO free output response, which will therefore now be given.

Assuming a constant output disturbance model for each process output, it is possible to generalize (3.106) and (3.110), which were derived directly from the model equations, onto the MIMO case – thus defining elements of the step responses and, in a recurrent way, free components of the predicted output trajectories. We shall consider a diagonal matrix $\mathbf{A}(z^{-1})$ only, denoting by $A^m(z^{-1})$ and $B^{m,j}(z^{-1})$ component polynomials of the polynomial matrices $\mathbf{A}(z^{-1})$ and $\mathbf{B}(z^{-1})$, which define a response of the $m$-th output on $n_u$ control inputs

$$y_m(k) = -\sum_{i=1}^{n_A} a_i^m y_m(k-i) + \sum_{j=1}^{n_u}\sum_{i=0}^{n_B} b_i^{m,j} u_j(k-1-i) \tag{3.114}$$

$m = 1, ..., n_y$ (it is assumed that each polynomial is of the same degree, $n_A^m = n_A$, $n_B^{m,j} = n_B$, without loss of generality). From the above dependence we immediately obtain a formula for elements of the step response, for each pair ($j$-th input)–($m$-th output),

$$s_k^{m,j} = - \sum_{i=1}^{\min\{k-1,n_A\}} a_i^m s_{k-i}^{m,j} + \sum_{i=0}^{\min\{k-1,n_B\}} b_i^{m,j} \tag{3.115}$$

$m = 1, ..., n_y$, $j = 1, ..., n_u$ (compare with (3.106)). Knowing the elements of a multivariable step response we can immediately formulate the corresponding dynamic matrix (see Section 3.2.1), and thus forced trajectories of the outputs in a prediction horizon.

Reasoning analogously as in the SISO case we can also easily obtain recurrent formulae for elements of free trajectories of the predicted outputs in the prediction horizon, which are equivalents of (3.110) for the MIMO case, in the form

$$\begin{aligned} y_m^0(k+p|k) = &- \sum_{i=1}^{\min\{n_A,p-1\}} a_i^m y_m^0(k+p-i|k) + \\ &- \sum_{i=\min\{n_A,p-1\}+1}^{n_A} a_i^m y_m(k+p-i) + \sum_{j=1}^{n_u} \left[ \sum_{i=0}^{\min\{n_B,p\}} b_i^{m,j} u_j(k-1) + \right. \\ &\left. + \sum_{i=\min\{n_B,p\}+1}^{n_B} b_i^{m,j} u_j(k-1+p-i) \right] + d_m(k) \end{aligned} \tag{3.116}$$

$p = 1, ..., N$, where

$$d_m(k) = y_m(k) - \left[ - \sum_{i=1}^{n_A} a_i^m y_m(k-i) + \sum_{j=1}^{n_u} \sum_{i=0}^{n_B} b_i^{m,j} u_j(k-1-i) \right]$$

are constant output disturbances calculated at current sampling instant $k$, $m = 1, ..., n_y$.

The design method presented above will be illustrated in Example 3.4 in Section 3.3.4, where a control system with the GPC algorithm, both in explicit and numerical versions, is demonstrated for an example two-input two-output process.

### 3.3.4 GPC Algorithm in Numerical Version

The formulations of quadratic optimization problems for different predictive control algorithms with linear process models differ in methods of calculation of the predicted values of the controlled outputs over the prediction horizon,

first of all the free component $\mathcal{Y}^0(k)$ of the predicted output trajectory. That is also the difference between the quadratic optimization problem in the numerical GPC algorithm, when compared to the analogous problem (3.69) in the DMC algorithm.

*The quadratic optimization problem* solved at each step of the GPC algorithm in the numerical version has the following form

$$\begin{aligned} \min_{\Delta\mathcal{U}(k)} \{ & \left\| [\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] - \mathbf{M}\,\Delta\mathcal{U}(k) \right\|^2_{\underline{\Psi}} + \|\Delta\mathcal{U}(k)\|^2_{\underline{\Lambda}} \} \\ \text{subj. to: } & -\Delta\mathcal{U}_{\max} \le \Delta\mathcal{U}(k) \le \Delta\mathcal{U}_{\max} \\ & \mathcal{U}_{\min} \le \mathcal{U}(k-1) + \mathbf{J}\Delta\mathcal{U}(k) \le \mathcal{U}_{\max} \\ & \mathcal{Y}_{\min} \le \mathcal{Y}^0(k) + \mathbf{M}\,\Delta\mathcal{U}(k) \le \mathcal{Y}_{\max} \end{aligned} \tag{3.117}$$

where the free output trajectory $\mathcal{Y}^0(k)$ is calculated from (3.113)

$$\mathcal{Y}^0(k) = \mathbf{F}\,\mathcal{Y}^{PG}(k) + \mathbf{G}^{PG}\Delta\mathcal{U}^{PG}(k)$$

All the values occurring in the above formulation were introduced previously in Section 3.2.4, where the numerical DMC algorithm was presented. The discussion given there concerning the formulation of the quadratic optimization problem in a standard form, feasibility of solutions *etc.*, remains of course valid also for the GPC algorithm.

**Example 3.4**
To illustrate a MIMO GPC algorithm, both in explicit and numerical versions, we shall present results of simulations for a two-dimensional linear process model given by the following transfer function matrix

$$\begin{bmatrix} Y_1(s) \\ Y_2(s) \end{bmatrix} = \begin{bmatrix} \dfrac{1}{1+0.7s} & \dfrac{5}{1+0.3s} \\ \dfrac{1}{1+0.5s} & \dfrac{2}{1+0.4s} \end{bmatrix} \begin{bmatrix} U_1(s) \\ U_2(s) \end{bmatrix}$$

representing a small-signal model of a continuous flow reactor at an operating point ([18], p. 143).

Assuming the sampling period $T_p = 0.03$ min we obtain the following discrete representation of the model

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} \dfrac{0.041951z^{-1}}{1-0.958048z^{-1}} & \dfrac{0.475812z^{-1}}{1-0.904837z^{-1}} \\ \dfrac{0.058235z^{-1}}{1-0.941764z^{-1}} & \dfrac{0.144513z^{-1}}{1-0.927743z^{-1}} \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

Converting elements of each of the two rows of the above matrix to a form with a common denominator we obtain the model in the form

$$\mathbf{A}(z^{-1})y(k) = \mathbf{B}(z^{-1})u(k-1)$$

where

$$\mathbf{A}(z^{-1}) = \begin{bmatrix} 1 + a_1^1 z^{-1} + a_2^1 z^{-2} & 0 \\ 0 & 1 + a_1^2 z^{-1} + a_2^2 z^{-2} \end{bmatrix}$$

$$= \begin{bmatrix} 1 - 1.862885 z^{-1} + 0.866877 z^{-2} & 0 \\ 0 & 1 - 1.869508 z^{-1} + 0.873715 z^{-2} \end{bmatrix}$$

$$\mathbf{B}(z^{-1}) = \begin{bmatrix} b_0^{1,1} + b_1^{1,1} z^{-1} & b_0^{1,2} + b_1^{1,2} z^{-1} \\ b_0^{2,1} + b_1^{2,1} z^{-1} & b_0^{2,2} + b_1^{2,2} z^{-1} \end{bmatrix}$$

$$= \begin{bmatrix} 0.041951 - 0.037959 z^{-1} & 0.475812 - 0.455851 z^{-1} \\ 0.058235 - 0.054027 z^{-1} & 0.144513 - 0.136097 z^{-1} \end{bmatrix}$$

The following values for the horizons were assumed: prediction horizon $N = 3$ and control horizon $N_u = 2$, also $N_1 = 1$ (time delays do not occur). Using (3.115) one calculates elements of step responses and finally the dynamic matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{0} \\ \mathbf{S}_2 & \mathbf{S}_1 \\ \mathbf{S}_3 & \mathbf{S}_2 \end{bmatrix} = \begin{bmatrix} 0.04195 & 0.47581 & 0 & 0 \\ 0.05824 & 0.14451 & 0 & 0 \\ 0.08214 & 0.90635 & 0.04195 & 0.47581 \\ 0.11308 & 0.27858 & 0.05824 & 0.14451 \\ 0.12065 & 1.29591 & 0.08214 & 0.90635 \\ 0.16473 & 0.40297 & 0.11308 & 0.27858 \end{bmatrix}$$

The formula for the output prediction, $\mathcal{Y}(k) = \mathcal{Y}^0(k) + \mathbf{M}\Delta\mathcal{U}(k)$, takes in the considered case the form

$$\begin{bmatrix} y_1(k+1|k) \\ y_2(k+1|k) \\ y_1(k+2|k) \\ y_2(k+2|k) \\ y_1(k+3|k) \\ y_2(k+3|k) \end{bmatrix} = \begin{bmatrix} y_1^0(k+1|k) \\ y_2^0(k+1|k) \\ y_1^0(k+2|k) \\ y_2^0(k+2|k) \\ y_1^0(k+3|k) \\ y_2^0(k+3|k) \end{bmatrix} + \mathbf{M} \begin{bmatrix} \triangle u_1(k|k) \\ \triangle u_2(k|k) \\ \triangle u_1(k+1|k) \\ \triangle u_2(k+1|k) \end{bmatrix}$$

The free output response $\mathcal{Y}^0(k)$ can be calculated in a recurrent way using (3.116):

$$y_1^0(k+1|k) = -a_1^1 y_1(k) - a_2^1 y_1(k-1) + b_0^{1,1} u_1(k-1) + \\ + b_1^{1,1} u_1(k-1) + b_0^{1,2} u_2(k-1) + b_1^{1,2} u_2(k-1) + d_1(k)$$

$$y_2^0(k+1|k) = -a_1^2 y_2(k) - a_2^2 y_2(k-1) + b_0^{2,1} u_1(k-1) + \\ + b_1^{2,1} u_1(k-1) + b_0^{2,2} u_2(k-1) + b_1^{2,2} u_2(k-1) + d_2(k)$$

$$y_1^0(k+2|k) = -a_1^1 y_1^0(k+1|k) - a_2^1 y_1(k) + b_0^{1,1} u_1(k-1) + \\ + b_1^{1,1} u_1(k-1) + b_0^{1,2} u_2(k-1) + b_1^{1,2} u_2(k-1) + d_1(k)$$

*etc.*, until

$$y_2^0(k+3|k) = -a_1^2 y_2^0(k+2|k) - a_2^2 y_2^0(k+1|k) + b_0^{2,1} u_1(k-1) + \\ + b_1^{2,1} u_1(k-1) + b_0^{2,2} u_2(k-1) + b_1^{2,2} u_2(k-1) + d_2(k)$$

where estimates of the disturbances are calculated as differences between the measured process outputs at sampling instant $k$ and model outputs for this instant calculated at the previous one, $k-1$:

$$d_1(k) = y_1(k) - [-a_1^1 y_1(k-1) - a_2^1 y_1(k-2) + \\ + b_0^{1,1} u_1(k-1) + b_1^{1,1} u_1(k-2) + b_0^{1,2} u_2(k-1) + b_1^{1,2} u_2(k-2)]$$

$$d_2(k) = y_2(k) - [-a_1^2 y_2(k-1) - a_2^2 y_2(k-2) + \\ + b_0^{2,1} u_1(k-1) + b_1^{2,1} u_1(k-2) + b_0^{2,2} u_2(k-1) + b_1^{2,2} u_2(k-2)]$$

It was assumed that $\mathbf{\Psi} = \mathbf{I}$ and $\mathbf{\Lambda} = \lambda \mathbf{I}$, with $\lambda = 0.075$. Simulations of the control system with the presented GPC algorithm were performed for the following two situations:

1. Unconstrained, explicit GPC algorithm. In this case the solution vector of the quadratic optimization problem is given analytically by the formula

$$\Delta \widehat{\mathcal{U}}(k) = \mathbf{K}[\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)]$$

where

$$\mathbf{K} = \begin{bmatrix} -0.04364 & 0.45453 & -0.20244 & 0.76346 & -0.32797 & 1.05657 \\ 0.90230 & 0.22863 & 0.44040 & 0.04689 & 0.02006 & -0.12258 \\ -0.12893 & -0.11711 & -0.15232 & 0.35308 & -0.15013 & 0.79786 \\ -1.27727 & -0.39466 & -0.19277 & -0.11970 & 0.78629 & 0.13457 \end{bmatrix}$$

while two first elements of the solution vector are used as the actual process control inputs at sampling instant $k$, $\triangle \hat{u}_1(k) = \triangle \hat{u}_1(k|k)$ and $\triangle \hat{u}_2(k) = \triangle \hat{u}_2(k|k)$.

2. Numerical GPC algorithm, with the following constraints on amplitudes and rates of change of the process inputs

$$-2.5 \leq u_1 \leq 2.5$$
$$-0.6 \leq u_2 \leq 0.6$$
$$-0.5 \leq \Delta u_1 \leq 0.5$$
$$-0.3 \leq \Delta u_2 \leq 0.3$$

where at each step of the algorithm a constrained quadratic optimization problem is solved.

Selected results of simulations with the unconstrained controller are presented in Fig. 3.28 (trajectories of set-points and controlled variables) and in Fig. 3.29 (trajectories of controller outputs/process inputs), while results of a simulation with the numerical controller are given in Fig. 3.30 and in Fig. 3.31, respectively. The trajectories are presented for a time interval corresponding to 250 algorithm steps (sampling periods). Within this interval, changes of the set-points took place at steps 10, 110 and 160, while at steps 50 and 200 changes of disturbances with an amplitude 0.2 occurred, at the first and second output, respectively.

In both situations good decoupling properties of the predictive controller can easily be seen. In the case of the unconstrained controller, approaching

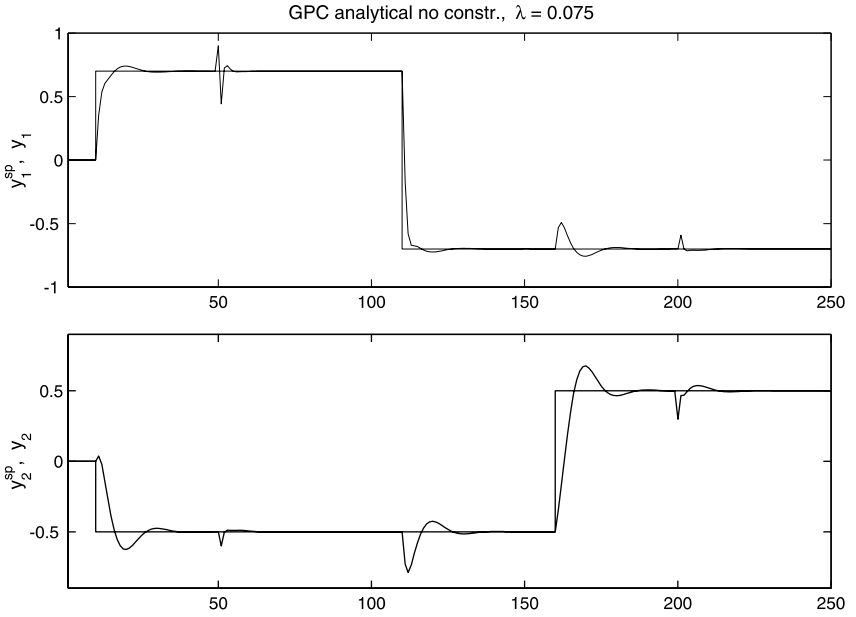


**Fig. 3.28.** Set-points and controlled outputs in the control system with unconstrained GPC controller (no constraints on process inputs)

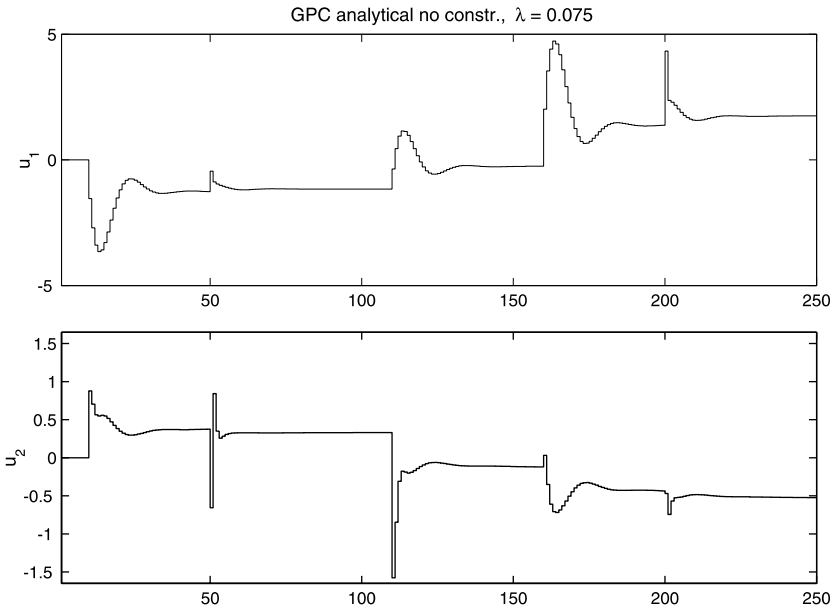


**Fig. 3.29.** Process inputs in the control system with unconstrained GPC controller

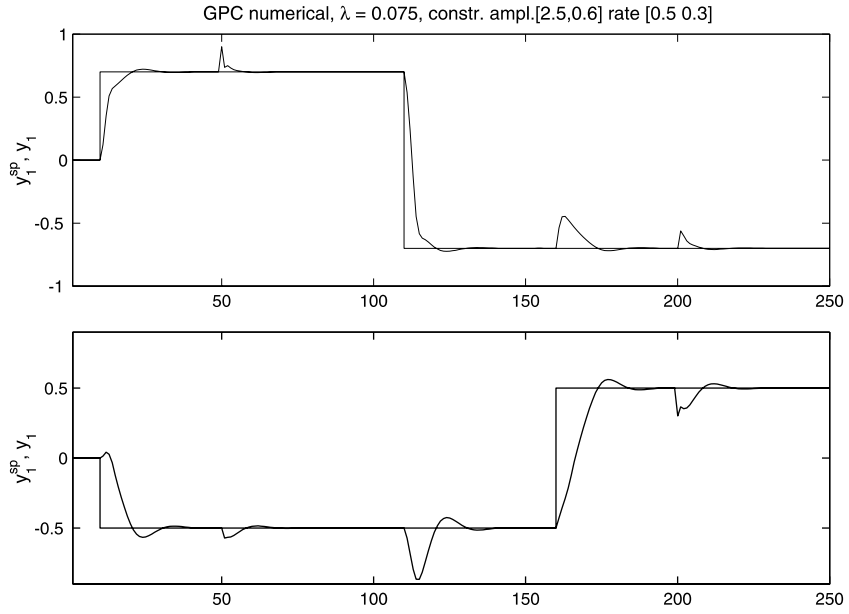


**Fig. 3.30.** Set-points and controlled outputs in the control system with numerical GPC controller (with constraints on process outputs)

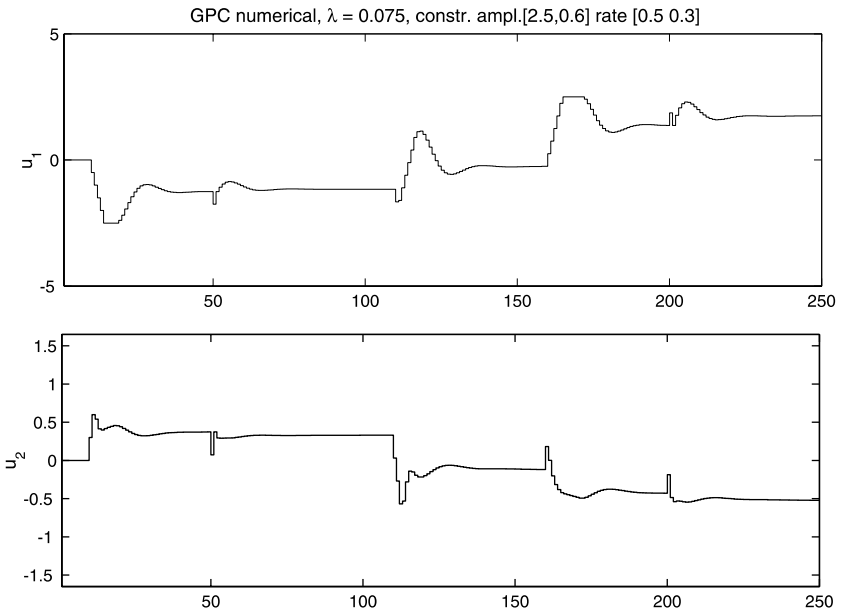


**Fig. 3.31.** Process inputs in the control system with numerical GPC controller

the set-points by the controlled variables (process outputs) is faster, but accompanied by larger amplitudes and changes of the process input signals. In a case with constraints the controller output trajectories are much more calm. Of course, smaller (but optimal for the considered case) amplitudes and rates of change of the process input signals result in longer rise times after step changes of the set-points and in weaker attenuation of the disturbances.

□

## 3.4 MPC with State-space Process Model

First generations of predictive control algorithms were based on linear input-output type models, in the form of pulse or step responses, and later in the form of discrete difference equations (see Section 3.1). Such models are dominant in process control applications, and thus in the area where the MPC algorithms were first derived and applied with a great success. These algorithms, particularly in a numerical version where at each sampling instant a quadratic programming problem is solved by a numerical optimization procedure, require quite a large amount of computations as compared to earlier on-line feedback controllers. Thus, it was most natural and easier to apply the MPC algorithms first for supervisory control of industrial processes, where longer sampling periods are usually used (see Chapter 1). Moreover,

appearance and wide application of electronic distributed control systems in industrial installations created a natural environment for implementations of more computationally demanding supervisory algorithms, such as feedback constraint control and optimization of set-points.

Introduction of predictive control algorithms with a process model in the form of a set of linear state-space equations occurred later, after further significant increase in computing power and reliability of computer control hardware, and at the same time when its dimensions and prices decreased. It became possible to use the MPC algorithms for direct digital control of small size processes, often isolated and well modeled by state-space equations, such as *e.g.*, electromechanical objects. Moreover, a process description in the form of state-space equations turned out to be more convenient for a theoretical analysis. In the 1990s the development and increase of popularity of predictive control algorithms based on state-space equations could be well recognized. For example, the *Model Predictive Control Toolbox* of the MATLAB® package, released in 1995, contains procedures for development and testing of MPC algorithms which can be based on either form of the process model: a step response or a set of linear state-space equations [99].

Formulation of the process model in the form of state-space equations is a more general approach, convenient for a theoretical analysis of MPC algorithms [82], especially for stability analysis. Moreover, it enables a natural generalization to cases with nonlinear models. However, in many industrial applications, where the input-output type modeling is dominant, a model in the state-space form can be not natural, especially in a typical situation when the full state vector is unavailable for measurement. Moreover, it is known that a MPC algorithm based on a description of the process by a system of linear state-space equations can have worse numerical properties, especially in cases of long prediction horizons and large dimensionality of the state vector. The MPC algorithm with the process model in the form of state-space equations will be referred to as *the MPCS algorithm* (*Model Predictive Control with State-space equations*) in this book, for notation simplicity.

### 3.4.1 Algorithms with Measured State

Consider the following form of the process state-space equations

$$x(k+1) = \mathbf{A}x(k) + \mathbf{B}u(k) + v(k) \tag{3.118}$$

$$y(k) = \mathbf{C}x(k) \tag{3.119}$$

where $x(k)$ is a *state vector* of a dimension $n_x$ which is assumed to be measured and $v(k)$ is a *state disturbance* which can also represent state modeling errors. We shall now derive formulae defining predicted values of the state variables assuming a *constant state disturbance prediction model* (which corresponds to modeling the disturbances as integrated white noises). That is,

$$v(k) = x(k) - [\mathbf{A}x(k-1) + \mathbf{B}u(k-1)] \tag{3.120}$$
$$v(k) = v(k+1|k) = \cdots = v(k+N-1|k) \tag{3.121}$$

Denoting by $x(k+p|k)$ a value of the state vector predicted at sampling instant $k$ for the future instant $k+p$ in the prediction horizon, and proceeding in a recurrent way, we obtain

$$\begin{aligned} x(k+1|k) &= \mathbf{A}x(k) + \mathbf{B}u(k|k) + v(k) \\ &= \mathbf{A}x(k) + \mathbf{B}(\triangle u(k|k) + u(k-1)) + v(k) \end{aligned}$$

$$\begin{aligned} x(k+2|k) &= \mathbf{A}x(k+1|k) + \mathbf{B}(\triangle u(k+1|k) + \triangle u(k|k) + u(k-1)) + v(k) \\ &= \mathbf{A}^2 x(k) + \mathbf{AB}(\triangle u(k|k) + u(k-1)) + \mathbf{A}v(k) \\ &\quad + \mathbf{B}(\triangle u(k+1|k) + \triangle u(k|k) + u(k-1)) + v(k) \\ &= \mathbf{A}^2 x(k) + (\mathbf{A}+\mathbf{I})\mathbf{B}\triangle u(k|k) + \mathbf{B}\triangle u(k+1|k) + \\ &\quad + (\mathbf{A}+\mathbf{I})\mathbf{B}u(k-1) + (\mathbf{A}+\mathbf{I})v(k) \end{aligned}$$

$$\begin{aligned} x(k+3|k) &= \mathbf{A}x(k+2|k) + \mathbf{B}(\triangle u(k+2|k) + \triangle u(k+1|k) + \\ &\quad + \triangle u(k|k) + u(k-1)) + v(k) \\ &= \mathbf{A}^3 x(k) + (\mathbf{A}^2 + \mathbf{A} + \mathbf{I})\mathbf{B}\triangle u(k|k) + (\mathbf{A}+\mathbf{I})\mathbf{B}\triangle u(k+1|k) + \\ &\quad + \mathbf{B}\triangle u(k+2|k) + (\mathbf{A}^2 + \mathbf{A} + \mathbf{I})\mathbf{B}u(k-1) + (\mathbf{A}^2 + \mathbf{A} + \mathbf{I})v(k) \\ &\quad \vdots \end{aligned}$$

$$\begin{aligned} x(k+N_u|k) &= \mathbf{A}x(k+N_u-1|k) + \mathbf{B}(\triangle u(k+N_u-1|k) + \cdots + \\ &\quad + \triangle u(k|k) + u(k-1)) + v(k) \\ &= \mathbf{A}^{N_u} x(k) + (\mathbf{A}^{N_u-1} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B}\triangle u(k|k) + \\ &\quad + (\mathbf{A}^{N_u-2} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B}\triangle u(k+1|k) + \\ &\quad + \cdots + \mathbf{B}\triangle u(k+N_u-1|k) + \\ &\quad + (\mathbf{A}^{N_u-1} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B}u(k-1) + (\mathbf{A}^{N_u-1} + \cdots + \mathbf{A} + \mathbf{I})v(k) \end{aligned}$$

$$\begin{aligned} x(k+N_u+1|k) &= \mathbf{A}x(k+N_u|k) + \mathbf{B}(\triangle u(k+N_u-1|k) + \cdots + \\ &\quad + \triangle u(k|k) + u(k-1)) \\ &= \mathbf{A}^{N_u+1} x(k) + (\mathbf{A}^{N_u} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B}\triangle u(k|k) + \\ &\quad + (\mathbf{A}^{N_u-1} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B}\triangle u(k+1|k) + \\ &\quad + \cdots + (\mathbf{A}+\mathbf{I})\mathbf{B}\triangle u(k+N_u-1|k) + \\ &\quad + (\mathbf{A}^{N_u} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B}u(k-1) + (\mathbf{A}^{N_u} + \cdots + \mathbf{A} + \mathbf{I})v(k) \\ &\quad \vdots \end{aligned}$$

$$
\begin{aligned}
x(k+N|k) = \mathbf{A}^N x(k) + (\mathbf{A}^{N-1} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B}\triangle u(k|k) + \\
+ (\mathbf{A}^{N-2} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B}\triangle u(k+1|k) + \cdots + \\
+ (\mathbf{A}^{N-N_u} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B}\triangle u(k+N_u-1|k) + \\
+ (\mathbf{A}^{N-1} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B}u(k-1) + (\mathbf{A}^{N-1} + \cdots + \mathbf{A} + \mathbf{I})v(k)
\end{aligned}
$$

Defining a vector $\mathcal{X}(k)$ and recalling the definition of $\triangle\mathcal{U}(k)$,

$$
\mathcal{X}(k) = \begin{bmatrix} x(k+1|k) \\ \vdots \\ x(k+N|k) \end{bmatrix}, \quad \Delta\mathcal{U}(k) = \begin{bmatrix} \triangle u(k|k) \\ \vdots \\ \triangle u(k+N_u-1|k) \end{bmatrix}
$$

the obtained formulae can be put together into a matrix form

$$
\begin{aligned}
\mathcal{X}(k) = & \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^{N_u} \\ \mathbf{A}^{N_u+1} \\ \vdots \\ \mathbf{A}^N \end{bmatrix} x(k) + \\
& + \begin{bmatrix} \mathbf{B} & \cdots & \mathbf{0} \\ (\mathbf{A}+\mathbf{I})\mathbf{B} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ (\mathbf{A}^{N_u-1} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B} & \cdots & \mathbf{B} \\ (\mathbf{A}^{N_u} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B} & \cdots & (\mathbf{A}+\mathbf{I})\mathbf{B} \\ \vdots & \ddots & \vdots \\ (\mathbf{A}^{N-1} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B} & \cdots & (\mathbf{A}^{N-N_u} + \cdots + \mathbf{A} + \mathbf{I})\mathbf{B} \end{bmatrix} \Delta\mathcal{U}(k) + \\
& + \begin{bmatrix} \mathbf{I} \\ \mathbf{A}+\mathbf{I} \\ \vdots \\ \mathbf{A}^{N_u-1} + \cdots + \mathbf{A} + \mathbf{I} \\ \mathbf{A}^{N_u} + \cdots + \mathbf{A} + \mathbf{I} \\ \vdots \\ \mathbf{A}^{N-1} + \cdots + \mathbf{A} + \mathbf{I} \end{bmatrix} \mathbf{B}u(k-1) + \begin{bmatrix} \mathbf{I} \\ \mathbf{A}+\mathbf{I} \\ \vdots \\ \mathbf{A}^{N_u-1} + \cdots + \mathbf{A} + \mathbf{I} \\ \mathbf{A}^{N_u} + \cdots + \mathbf{A} + \mathbf{I} \\ \vdots \\ \mathbf{A}^{N-1} + \cdots + \mathbf{A} + \mathbf{I} \end{bmatrix} v(k)
\end{aligned}
$$

*i.e.*, denoting matrices in the above equation by $\widetilde{\mathbf{A}}$, $\mathbf{M}_x$ and $\mathbf{V}$,

$$
\mathcal{X}(k) = \widetilde{\mathbf{A}}x(k) + \mathbf{M}_x\Delta\mathcal{U}(k) + \mathbf{V}\mathbf{B}u(k-1) + \mathbf{V}v(k) \tag{3.122}
$$

Observe that $N_1 = 1$ was assumed, without loss of generality, as cases with $N_1 > 1$ can easily be obtained by removing an appropriate number of first

rows from the presented prediction equations. Defining the matrix $\widetilde{\mathbf{C}}$ of dimension $N \cdot n_y \times N \cdot n_x$

$$\widetilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{C} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{C} \end{bmatrix}$$

the equation describing the trajectory of predicted outputs is obtained

$$\begin{aligned} \mathcal{Y}^{pred}(k) &= \widetilde{\mathbf{C}}\mathcal{X}(k) \\ &= \left[\widetilde{\mathbf{C}}\widetilde{\mathbf{A}}x(k) + \widetilde{\mathbf{C}}\mathbf{V}\mathbf{B}u(k-1) + \widetilde{\mathbf{C}}\mathbf{V}v(k)\right] + \widetilde{\mathbf{C}}\mathbf{M}_x \Delta\mathcal{U}(k) \\ &= \mathcal{Y}^0(k) + \Delta\mathcal{Y}(k) \end{aligned} \tag{3.123}$$

or equivalently, as separate formulae for subsequent future sampling instants:

$$\begin{aligned} y(k+p|k) = \mathbf{C}\mathbf{A}^p x(k) + \mathbf{C}\,[\sum_{j=0}^{p-1}\mathbf{A}^j]\,\mathbf{B}u(k-1) + \mathbf{C}\,[\sum_{j=0}^{p-1}\mathbf{A}^j]\,v(k) + \\ + \mathbf{C}\sum_{i=0}^{p-1}[\sum_{j=0}^{p-1-i}\mathbf{A}^j]\,\mathbf{B}\triangle u(k+i|k), \quad \text{for } p \le N_u \end{aligned}$$

$$\begin{aligned} y(k+p|k) = \mathbf{C}\mathbf{A}^p x(k) + \mathbf{C}\,[\sum_{j=0}^{p-1}\mathbf{A}^j]\,\mathbf{B}u(k-1) + \mathbf{C}\,[\sum_{j=0}^{p-1}\mathbf{A}^j]\,v(k) + \\ + \mathbf{C}\sum_{i=0}^{N_u-1}[\sum_{j=0}^{p-1-i}\mathbf{A}^j]\mathbf{B}\triangle u(k+i|k), \quad \text{for } p > N_u \end{aligned}$$

where the last sums are elements of a forced component of the predicted outputs trajectory, $\Delta\mathcal{Y}(k) = \widetilde{\mathbf{C}}\mathbf{M}_x\Delta\mathcal{U}(k)$.

Certainly, the equality $\widetilde{\mathbf{C}}\mathbf{M}_x = \mathbf{M}$ must hold, where $\mathbf{M}$ is the dynamic matrix (3.39). Note that the matrix $\widetilde{\mathbf{C}}\mathbf{M}_x$ is defined by a fairly complicated formula, but its elements are simply coefficients of the step responses. Note also that the formulae given above become numerically more complicated for larger dimensionalities $n_x$ of the state vector and for longer prediction horizons. It is particularly inconvenient when the number of the controlled outputs $n_y$ is much smaller than $n_x$.

Assuming the same form of the cost function as in the DMC algorithm, see (3.40), and lack of inequality constraints on process inputs and outputs, we obtain a solution to the MPCS controller optimization problem as a solution to the set of linear equations (3.42). Its explicit, analytical form is given by (3.43), *i.e.*,

$$\begin{aligned} \Delta\widehat{\mathcal{U}}(k) &= \mathbf{K}[\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] \\ &= \mathbf{K}[\mathcal{Y}^{sp}(k) - \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}x(k) - \widetilde{\mathbf{C}}\mathbf{V}\mathbf{B}u(k-1) - \widetilde{\mathbf{C}}\mathbf{V}v(k)] \end{aligned} \tag{3.124}$$

where the matrix $\mathbf{K}$, although easily given by (3.44), should be calculated with great care, see the discussion and reasoning leading to the numerically better formula (3.65).

Only elements of the optimal control vector corresponding to current sampling instant are applied to the process, this results in the following explicit unconstrained MPCS control law:

$$\triangle\hat{u}(k) = \overline{\mathbf{K}}_1[\mathcal{Y}^{sp}(k) - \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}x(k) - \widetilde{\mathbf{C}}\mathbf{V}\mathbf{B}u(k-1) - \widetilde{\mathbf{C}}\mathbf{V}v(k)] \tag{3.125}$$

where $\overline{\mathbf{K}}_1$ is the first row sub-matrix of the matrix $\mathbf{K}$, see (3.46).

Assuming constant set-point over the prediction horizon, see (3.51), and exploiting structures of matrices $\widetilde{\mathbf{A}}$, $\mathbf{V}$ and $\widetilde{\mathbf{C}}$, the control law (3.125) can be written in the form

$$\triangle\hat{u}(k) = \sum_{p=1}^{N} \mathbf{K}_{1,p}\,[y^{sp}(k) - \mathbf{C}\mathbf{A}^p x(k) - \mathbf{C}\mathbf{V}_p\,\mathbf{B}u(k-1) - \mathbf{C}\mathbf{V}_p\,v(k)] \tag{3.126}$$

where $\mathbf{K}_{1,p}$ are sub-matrices (in the SISO case scalars) of $\overline{\mathbf{K}}_1$ (see (3.46), for $N_1 = 1$),

$$\overline{\mathbf{K}}_1 = [\,\mathbf{K}_{1,1}\;\;\mathbf{K}_{1,2}\;\;\cdots\;\;\mathbf{K}_{1,N}\,]$$

and $\mathbf{V}_p$ are sub-matrices of $\mathbf{V}$,

$$\mathbf{V}_p = \sum_{j=0}^{p-1} \mathbf{A}^j, \quad p = 1, ..., N$$

Defining $\mathbf{K}^e$ as in (3.53),

$$\mathbf{K}^e = \sum_{p=1}^{N} \mathbf{K}_{1,p}$$

the control law (3.126) can be written as

$$\triangle\hat{u}(k) = \mathbf{K}^e y^{sp}(k) - \sum_{p=1}^{N} \mathbf{K}_{1,p}\,[\mathbf{C}\mathbf{A}^p x(k) + \mathbf{C}\mathbf{V}_p\,(\mathbf{B}u(k-1) + v(k))] \tag{3.127}$$

where disturbance estimate $v(k)$ is given by (3.120),

$$v(k) = x(k) - [\mathbf{A}x(k-1) + \mathbf{B}u(k-1)] \tag{3.128}$$

The structure of the obtained MPCS control law is illustrated in Fig. 3.32. It is a linear feedback from the current (measured) state $x(k)$ and the last process input value $u(k-1)$. The structure of the state disturbance estimate is shown in Fig. 3.32 as well, within the dashed-line block.

The process model can be often formulated in the form of difference equations, as it was described in the case of the GPC algorithm, see (3.84). Assume the model in this form (with disturbances omitted)

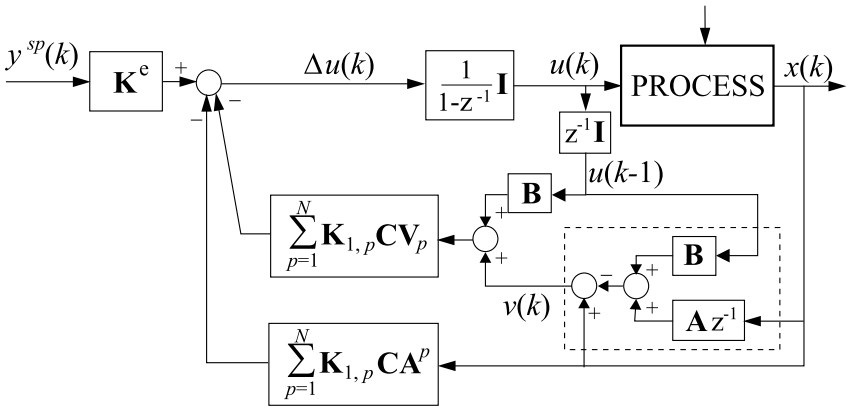


**Fig. 3.32.** Structure of the unconstrained explicit MPCS control law, with measured state and constant state disturbance prediction (exact output equation assumed)

$$\mathbf{A}(z^{-1})y(k) = \mathbf{B}(z^{-1})u(k-1) \tag{3.129}$$

where polynomial matrices $\mathbf{A}(z^{-1})$ and $\mathbf{B}(z^{-1})$ are given by (3.83a) and (3.83b), respectively. A state-space representation of this process model can easily be constructed, defining the following state vector consisting of past process outputs and inputs:

$$x(k) = [y(k)^T \; y(k-1)^T \cdots \; y(k-n_A+1)^T \; u(k-1)^T \cdots \; u(k-n_B+1)^T]^T \tag{3.130}$$

Matrices $A, B, C$ of the state-space model (3.118)-(3.119) corresponding to the state definition (3.130) are as follows

$$A = \begin{bmatrix} -\mathbf{A}_1 & -\mathbf{A}_2 & \cdots & -\mathbf{A}_{n_A-1} & -\mathbf{A}_{n_A} & \mathbf{B}_1 & \cdots & \mathbf{B}_{n_B-1} & \mathbf{B}_{n_B} \\ \mathbf{I}_y & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_y & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_y & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{I}_u & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I}_u & \mathbf{0} \end{bmatrix}$$

$$B = [\mathbf{B}_0^T \;\; \mathbf{0} \;\; \mathbf{0} \;\; \cdots \;\; \mathbf{0} \;\; \mathbf{I}_u \;\; \mathbf{0} \;\; \cdots \;\; \mathbf{0}]^T$$

$$C = [\mathbf{I}_y \;\; \mathbf{0} \;\; \mathbf{0} \;\; \cdots \;\; \mathbf{0} \;\; \mathbf{0} \;\; \mathbf{0} \;\; \cdots \;\; \mathbf{0}]$$

where $\dim \mathbf{I}_y = n_y \times n_y$, $\dim \mathbf{I}_u = n_u \times n_u$. The presented approach to state-space modeling leads often to a non-minimal state representation, but it is an important practical case when the state vector is measured.

When the entire state is measured, the output equation (3.119) is usually a precise one, the output variables being a part or a linear combination of the state variables. However, it may happen that the original output equation is nonlinear, as it is *e.g.*, in the case of a nonlinear state-space model of a polymerization reactor in Example 3.8 in Section 3.5. Using then an approximate, linearized form of the output equations introduces additional modeling error. Using the additive state disturbances only is then not sufficient for offset-free feedback control. To eliminate the offset, a bias signal must be added, such that at the equilibrium point (in the steady-state) values of the original output equation, $y = g(x)$, and the linearized one, $y = \mathbf{C}x + d$, are equal, *i.e.*, $g(x^{ss}) = \mathbf{C}x^{ss} + d(y^{sp})$, where $x^{ss}$ is the equilibrium state value corresponding to the set-point value $y^{sp}$. As indicated, the bias value depends on the set-point value, therefore it should be adopted on-line to the set-point changes. A simple way to do it is to add to the set-point the required difference, given by

$$d(k) = -\mathbf{C}x(k) + g(x(k)) \tag{3.131}$$

An alternative approach is to treat the difference between the original and model output equations, resulting from modeling inaccuracies, as unmeasured additive output disturbances, with constant predictions over the prediction horizon (as it was introduced in the DMC algorithm). The disturbance model would than be

$$\begin{aligned} d(k) &= y(k) - y(k|k-1) \\ &= y(k) - \mathbf{C}x(k|k-1) \\ &= y(k) - \mathbf{C}[\mathbf{A}x(k-1) + \mathbf{B}u(k-1) + v(k-1)] \end{aligned} \tag{3.132}$$

where $y(k) = g(x(k))$, and $d(k)$ should be added to the prediction equations (3.123). This results in a change of the free output trajectory to the form

$$\mathcal{Y}^0(k) = \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}x(k) + \widetilde{\mathbf{C}}\mathbf{V}\mathbf{B}u(k-1) + \widetilde{\mathbf{C}}\mathbf{V}v(k) + \widetilde{\mathbf{I}}_y d(k) \tag{3.133}$$

where $\widetilde{\mathbf{I}}_y = [\mathbf{I}_y\ \mathbf{I}_y\ \cdots \mathbf{I}_y]^T$, $\dim \widetilde{\mathbf{I}}_y = N \cdot n_y \times n_y$ (for $N_1 = 1$). The resulting explicit, unconstrained MPCS control law then is

$$\triangle\hat{u}(k) = \mathbf{K}^e[y^{sp}(k) - d(k)] - \sum_{p=1}^{N} \mathbf{K}_{1,p}\left[\mathbf{C}\mathbf{A}^p x(k) + \mathbf{C}\mathbf{V}_p\left(\mathbf{B}u(k-1) + v(k)\right)\right] \tag{3.134}$$

compare with (3.127).

**Example 3.5**

In order to illustrate the design of a MPCS controller with measured state, such a design will be performed for the process from Examples 3.1 and 3.2, described by a discrete difference equation (3.102)

$$(1 - 0.2676z^{-1})y(k) = (0.1989z^{-2} + 0.2552z^{-3})u(k-1) \tag{3.135}$$

As previously, it is assumed that $N_1 = 3$, $N_u = 3$, $N = 6$ and $\mathbf{\Psi}(p)=\mathbf{I}$, $\mathbf{\Lambda}(p)=\lambda\mathbf{I}$.

The state will now be defined in the form (3.130), as follows

$$x(k) = [\, y(k)\ u(k-1)\ u(k-2)\ u(k-3)\,]$$

Using this definition, the following set of state and output equations is the representation of the discrete transfer function (3.135)

$$x(k+1) = \begin{bmatrix} 0.2676 & 0 & 0.1989 & 0.2552 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} u(k) + v(k)$$

$$y(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} x(k)$$

where $v(k)$ are state disturbances introduced as in (3.118), which are used to counteract influences of outer disturbances and modeling errors. In can be easily checked that the obtained model is one of possible minimal state-space representations (compare with the state-space model in the next example).

For the assumed process the unconstrained MPCS control law (3.127), (3.128) was computed and the control system was simulated, in the structure taking into account constraints both on the amplitude and rate of change of the process input ($u(k) \leq 2$, $|\triangle u(k)| \leq 0.5$), by projection of the controller output signal on the feasible set and with the anti-windup structure added, similarly as it was shown in the controller structure presented in Fig. 3.14 for the DMC controller. Tests were conducted for the process output disturbed by an additive step signal changing its value from 0 to 0.45 at sampling instant $k = 2$, for the model equal to the process and for differences in the gain of the process and its model.

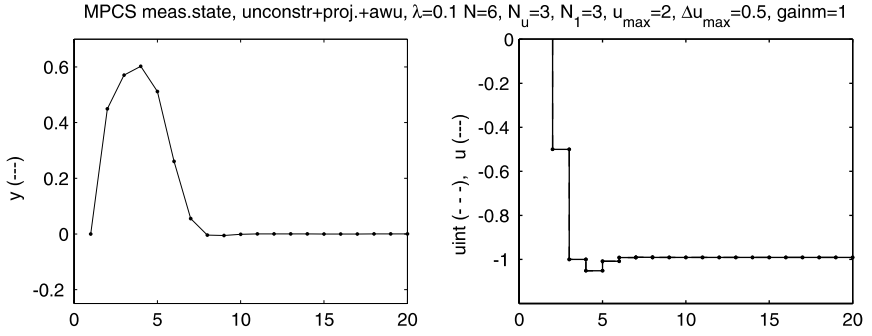


**Fig. 3.33.** Response to a disturbance step in the control system with explicit MPCS controller, with measured state and without modeling errors

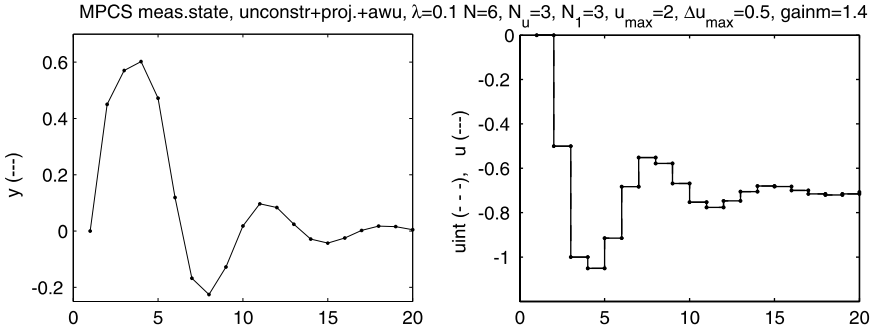


**Fig. 3.34.** Response to a disturbance step in the control system with explicit MPCS controller with measured state, with the gain in the process (but not in the model) increased by 40%

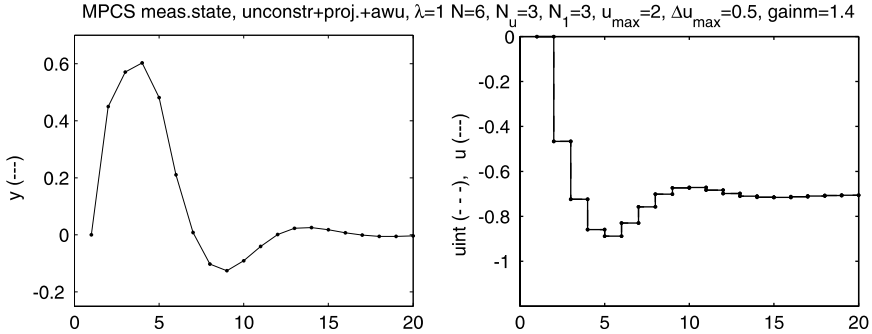


**Fig. 3.35.** Response to a disturbance step in the control system with explicit MPCS controller with measured state, with the gain in the process (but not in the model) increased by 40% and increased penalty for control input moves

Three selected figures are presented which illustrate the obtained results. Fig. 3.33 presents the case without modeling errors. The obtained trajectory is fast, almost identical to that in Fig. 3.17 for the DMC controller – certainly, the first two control input moves are identical due to activity of the rate of change constraint ($|\triangle u(k)| \leq 0.5$), therefore differences can be seen in the further part of the trajectories only. Fig. 3.34 presents trajectories with the process gain increased by 40%, but the model unchanged. The control system remains stable, but the response becomes oscillatory. To damp oscillations the value of $\lambda$ was increased to 1, the resulting trajectories are presented in Fig. 3.35. □

### 3.4.2 Algorithms with Estimated State

The entire state vector is often unavailable for measurements, the cases with measured state consisting of past process outputs and inputs, as discussed in the previous section, constitute a special class, see also *e.g.*, [82].

In general, it may be necessary to use a *state observer*. For a process described by the state-space and output equations (3.118)-(3.119), the state observer (Luenberger observer) can be presented in the following form

$$\hat{x}(k+1|k) = \mathbf{A}\hat{x}(k|k-1) + \mathbf{B}u(k) + \mathbf{L}_p[y(k) - \mathbf{C}\hat{x}(k|k-1)] \tag{3.136}$$

where $\hat{x}(k|k-1)$ denotes an estimate of the state vector $x(k)$ evaluated on the basis of information available at the previous sampling instant $k-1$, while $\mathbf{L}_p$ is a gain matrix defining the observer dynamics. Observability of a linear dynamic system described by matrices $\mathbf{A}$ and $\mathbf{C}$ is a sufficient condition for any positioning of poles of the state matrix $\mathbf{A} - \mathbf{L}_p\mathbf{C}$ of the observer (3.136), performed by a suitable selection of the matrix $\mathbf{L}_p$, see *e.g.*, [45, 3]. The matrix $\mathbf{A} - \mathbf{L}_p\mathbf{C}$ is also a matrix of the dynamics of the estimation error $x(k) - \hat{x}(k|k-1)$.

The observer given by (3.136) is called a *predictive observer* or a *predictive estimator* [45], as it generates a state estimate for the next sampling instant on the basis of the current state estimate $\hat{x}(k|k-1)$ evaluated at the previous instant and a current measurement information. Thus, at the current sampling instant we have only a state estimate $\hat{x}(k|k-1)$ evaluated on the basis of the past measurement information $y(k-1)$, that is on the basis of delayed information. If the time of calculation of the control signal is significantly shorter than the sampling period, then it is better to have at the current sampling instant a state estimate based on the current information. A state observer which fulfills this requirement is called *current* [45] and is usually used as a standard state observer (state estimator). Its equations are as follows:

$$\hat{x}(k|k) = \hat{x}(k|k-1) + \mathbf{L}[y(k) - \mathbf{C}\hat{x}(k|k-1)] \tag{3.137}$$

$$\hat{x}(k+1|k) = \mathbf{A}\hat{x}(k|k) + \mathbf{B}u(k) \tag{3.138}$$

and after inserting the first equation into the second one we obtain

$$\hat{x}(k+1|k) = \mathbf{A}\hat{x}(k|k-1) + \mathbf{B}u(k) + \mathbf{AL}[y(k) - \mathbf{C}\hat{x}(k|k-1)] \tag{3.139}$$

Note that the above equation is identical with (3.136), provided $\mathbf{L}_p = \mathbf{AL}$. Design of the dynamics of the current observer can be conducted calculating first values of the gain matrix $\mathbf{L}_p$ according to the chosen pole positions of a predictive observer, and next calculating elements of the matrix $\mathbf{L}$ [45].

Not only original process state variables, but also additional state variables which represent *dynamics of uncontrolled inputs* (disturbances with a significant deterministic component) should be estimated. According to the *internal model principle* a compensation of disturbances which do not decay

in an autonomous way is possible only if the model of their dynamics is a properly used element of the controller dynamics, and state estimation is an integral part of the controller, see *e.g.*, [154, 3, 52].

The augmented state-space model of a process including its persistent disturbances affecting the state can be generally formulated in the following form, see *e.g.*, [82],

$$\begin{bmatrix} x(k+1) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{X_v} \\ \mathbf{0} & \mathbf{A_v} \end{bmatrix} \begin{bmatrix} x(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} u(k)$$

where $v(k)$ is a state vector of the disturbances. The disturbances described in this way can be of various nature, *e.g.*, can be periodic. A suitable observer should be designed for the augmented model. For this topic, the reader is referred to the literature, in particular to the book [82], devoted mainly to predictive control with state-space process models.

An important case is modeling the disturbances as affecting additively the process output, then the process dynamics is described as follows

$$x(k+1) = \mathbf{A}x(k) + \mathbf{B}u(k) \tag{3.140}$$

$$y(k) = \mathbf{C}x(k) + d(k) \tag{3.141}$$

Most important, from a practical point of view, is the case with disturbances $d(k)$ modeled as integrated white noise added to the process output, which results in a constant output disturbance prediction – as it was originally assumed in the DMC algorithm and discussed in the GPC algorithm. Dynamics of disturbances is then of the form

$$d(k+1) = d(k) + \epsilon(k) \tag{3.142}$$

where $\epsilon(k)$ is a zero-mean white noise. Combined equations of the process and disturbance dynamics are then

$$\begin{bmatrix} x(k+1) \\ d(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} x(k) \\ d(k) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} u(k) + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \epsilon(k)$$

$$y(k) = \begin{bmatrix} \mathbf{C} & \mathbf{I} \end{bmatrix} \begin{bmatrix} x(k) \\ d(k) \end{bmatrix}$$

and the observer equations are

$$\begin{bmatrix} \hat{x}(k|k) \\ \hat{d}(k|k) \end{bmatrix} = \begin{bmatrix} \hat{x}(k|k-1) \\ \hat{d}(k|k-1) \end{bmatrix} + \begin{bmatrix} \mathbf{L} \\ \mathbf{L}_d \end{bmatrix} [y(k) - \begin{bmatrix} \mathbf{C} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{x}(k|k-1) \\ \hat{d}(k|k-1) \end{bmatrix}] \tag{3.143a}$$

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \hat{d}(k+1|k) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{x}(k|k) \\ \hat{d}(k|k) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} u(k) \tag{3.143b}$$

where $\widetilde{\mathbf{L}} = [\mathbf{L}^T \ \mathbf{L}_d^T]^T$ is the observer gain matrix. This results in the combined form of the observer dynamics

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \hat{d}(k+1|k) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{x}(k|k-1) \\ \hat{d}(k|k-1) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} u(k) + \\ + \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{L} \\ \mathbf{L}_d \end{bmatrix} [y(k) - \begin{bmatrix} \mathbf{C} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{x}(k|k-1) \\ \hat{d}(k|k-1) \end{bmatrix}] \tag{3.144}$$

On the other hand, an estimate of disturbances affecting the process output at sampling instant $k$, $\hat{d}(k|k)$, should be equal to the difference of the measured output value and the output value predicted for this instant one sample behind, *i.e.*,

$$\hat{d}(k|k) = y(k) - \mathbf{C}\hat{x}(k|k-1)$$

It follows from the assumed disturbance model that a current disturbance estimate is the optimal prediction for future sampling instants,

$$\hat{d}(k+1|k) = \hat{d}(k|k) = y(k) - \mathbf{C}\hat{x}(k|k-1)$$

Such an estimate will be obtained by the observer (3.144) if

$$\mathbf{L}_d = \mathbf{I} \tag{3.145}$$

Moreover, if we assume $\mathbf{L} = \mathbf{0}$, then equations of the observer dynamics reduce to the following form

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \hat{d}(k+1|k) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ -\mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{x}(k|k-1) \\ \hat{d}(k|k-1) \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} u(k) + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} y(k)$$

Eigenvalues of the state matrix of such an observer which correspond to the states of the disturbance vector are equal to zero, while the remaining eigenvalues are equal to the eigenvalues of the matrix $\mathbf{A}$. This corresponds to the estimation of disturbances in a finite number of steps (a deadbeat-type estimation). The presented design method of a controller (observer), see [100, 82], ensures zero control errors in steady-states and a fast estimation. However, it significantly constrains a choice of pole positions of the observer dynamics. Particularly, it can only be used for stable processes. However, it turns out that it is only a special case of design of a MPCS predictive control system ensuring zero steady-state control errors, for the constant output disturbance model. This will now be discussed.

Assume that an MPCS control system with an observer (3.144) is asymptotically stable and that there exists a feasible process input which ensures that in a steady-state the process output values are equal to the assumed setpoint values $y^{sp}$ and that the prediction horizon is sufficiently long to enable the predicted outputs to stabilize safely at $y^{sp}$, after expected changes of setpoints or disturbances. Let us denote by $y_\infty$, $x_\infty$ and $d_\infty$ steady-state limits of values of the outputs, process state estimates and disturbance estimates, namely $y_\infty = \lim_{k\to\infty} y(k)$, $x_\infty = \lim_{k\to\infty} \hat{x}(k+1|k)$, $d_\infty = \lim_{k\to\infty} \hat{d}(k+1|k)$ (in a steady-state estimation errors converged to zero). From the bottom part

of (3.144), which corresponds to estimates of the disturbances, we obtain in a steady-state

$$\mathbf{L}_d[y_\infty - (\mathbf{C}x_\infty + d_\infty)] = 0$$

Thus, if only the matrix $\mathbf{L}_d$ is non-singular,

$$\det \mathbf{L}_d \neq 0 \tag{3.146}$$

then $y_\infty = \mathbf{C}x_\infty + d_\infty$. On the other hand, a predictive controller minimizes the cost function which in a steady-state takes the following value

$$J_\infty = \sum_{p=N_1}^{N} \|y^{sp} - (\mathbf{C}x_\infty + d_\infty)\|^2_{\mathbf{\Psi}(p)}$$

Since we assumed the existence of process control inputs which ensure that in the steady-state the process outputs are equal to $y^{sp}$, *i.e.*, the smallest possible value (zero) of the cost function is achieved, then $y^{sp} = \mathbf{C}x_\infty + d_\infty$ must hold. On the other hand, $\mathbf{C}x_\infty + d_\infty = y_\infty$ if the condition (3.146) holds. Thus, the equality $y_\infty = y^{sp}$ should be true, *i.e.*, the equality of steady-state values of the process outputs and values of the set-points – resulting in steady-state control errors equal to zero. In real situations the process model is often uncertain, not known accurately. Elements of the matrix of the observer gains are then usually treated as additional tuning parameters of the controller during the design phase and the possibility of its proper, less constrained selection can be useful.

If the state-space model (3.140)-(3.141) is assumed, than the state predictions must be appropriately modified, as there are no disturbances directly affecting the state equations. As a result, the term involving $v(k)$ will disappear from the state predictions, but the disturbance estimate $\hat{d}(k|k)$ will be added to the output equations. Thus, the formula for the state prediction will now be, compare with (3.122),

$$\mathcal{X}(k) = \widetilde{\mathbf{A}}\hat{x}(k|k) + \mathbf{M}_x \Delta\mathcal{U}(k) + \mathbf{V}\mathbf{B}u(k-1) \tag{3.147}$$

and output prediction will be given by, compare with(3.123),

$$\begin{aligned}\mathcal{Y}^{pred}(k) &= \widetilde{\mathbf{C}}\mathcal{X}(k) + \widetilde{\mathbf{I}}_y \hat{d}(k|k) \\ &= \left[\widetilde{\mathbf{C}}\widetilde{\mathbf{A}}\hat{x}(k|k) + \widetilde{\mathbf{C}}\mathbf{V}\mathbf{B}u(k-1) + \widetilde{\mathbf{I}}_y \hat{d}(k|k)\right] + \widetilde{\mathbf{C}}\mathbf{M}_x \Delta\mathcal{U}(k) \\ &= \mathcal{Y}^0(k) + \Delta\mathcal{Y}(k)\end{aligned} \tag{3.148}$$

where $\widetilde{\mathbf{I}}_y = [\mathbf{I}_y\ \mathbf{I}_y\ \cdots \mathbf{I}_y]^T$, $\dim \widetilde{\mathbf{I}}_y = N \cdot n_y \times n_y$ (for $N_1 = 1$ ) and $\hat{x}(k|k)$ and $\hat{d}(k|k)$ are state and disturbance estimates obtained from the joint current observer (3.143a)-(3.143b).

Under the same assumptions and reasoning as before, solving the controller unconstrained optimization problem, one arrives finally at the following *explicit unconstrained MPCS control law*, analogous to (3.127),

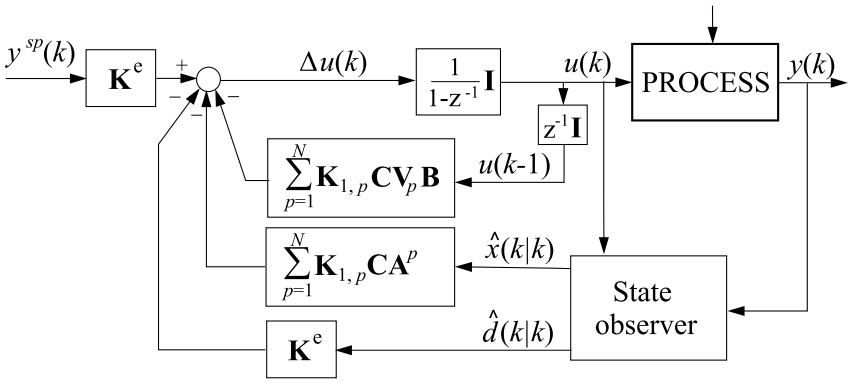


**Fig. 3.36.** Structure of the unconstrained, explicit MPCS control law, with estimated state and output disturbance

$$\triangle\hat{u}(k) = \mathbf{K}^e[y^{sp}(k) - \hat{d}(k|k)] - \sum_{p=1}^{N} \mathbf{K}_{1,p}\left[\mathbf{C}\mathbf{A}^p\hat{x}(k|k) + \mathbf{C}\mathbf{V}_p\mathbf{B}u(k-1)\right] \tag{3.149}$$

The obtained control law is a linear feedback from the estimated process state, estimated output disturbances, and also from the last process input values, which is illustrated in Fig. 3.36.

When there are constraints on amplitudes or rates of change of the process input signals, the presented control structures with the unconstrained explicit feedback control laws should be appropriately augmented to take the constraints into account, in the same way as it was presented for the control structures with the unconstrained DMC algorithm.

In cases with stochastic disturbances affecting the process states and outputs, it is common to use a Kalman filter for the state estimation, also for a joint state including disturbances of a deterministic type, *e.g.*, step disturbances affecting inputs or outputs of the process model, see *e.g.*, [100, 45, 3, 82].

**Example 3.6**

In order to illustrate the design of a MPCS controller with state observer, the process from Example 3.5 will further be considered, described by the discrete difference equation (3.135).

As in the previous examples, it is assumed that $N_1 = 3$, $N_u = 3$, $N = 6$ and $\mathbf{\Psi}(p)= \mathbf{I}$, $\mathbf{\Lambda}(p)=\lambda\mathbf{I}$ and constraints on the amplitude and rate of change of the process input are applied, $|u(k)| \leq 2$, $|\triangle u(k)| \leq \triangle 0.5$. The constraints are taken into account by projection of the controller output onto the feasible set, the anti-windup loop is also used.

As can easily be proven, the following set of state and output equations is one of possible minimal representations corresponding to the discrete transfer function (3.135)

$$x(k+1) = \begin{bmatrix} 0.2676 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 0.61685 & 0.1989 & 0 & 0 \end{bmatrix} x(k) + d(k)$$

In the above output equation a disturbance $d(k)$ was added to the output, since a constant output disturbance prediction is assumed, as it was in previous examples with the DMC and GPC algorithms.

Since the last three components of the state vector correspond to the time delay, then the observer of the combined state (3.143a)-(3.143b) with the observer matrix $\widetilde{\mathbf{L}}$ in the form

$$\widetilde{\mathbf{L}} = [\, \mathbf{L}^T \ \mathbf{L}_d^T \,]^T = [\, l_1 \ 0 \ 0 \ 0 \ l_d \,]^T$$

is assumed.

For the assumed process and disturbance models the unconstrained MPCS control law was computed and the control system was simulated. Tests were conducted for a change in the disturbance value from 0 to 0.45 at sampling instant $k = 2$, for the model equal to the process and for cases with differences between the process and its model, in the gain and in the structure. Four selected figures are presented which illustrate the obtained results, where dashed lines present trajectories of the disturbance estimate $\hat{d}(k|k)$.

Fig. 3.37 presents the case without modeling error and with observer parameters $l_1 = 0$ and $l_d = 1$, which correspond to the eigenvalues of the observer state matrix $\alpha_1 = 0.2676$ and $\alpha_5 = 0$ (the remaining eigenvalues are structurally equal to zero). The obtained trajectory is fast, practically identical to that in Fig. 3.33 obtained in the case with measured state. The control system was also tested with $l_1 = -0.305$ and $l_d = 1.188$, corresponding to

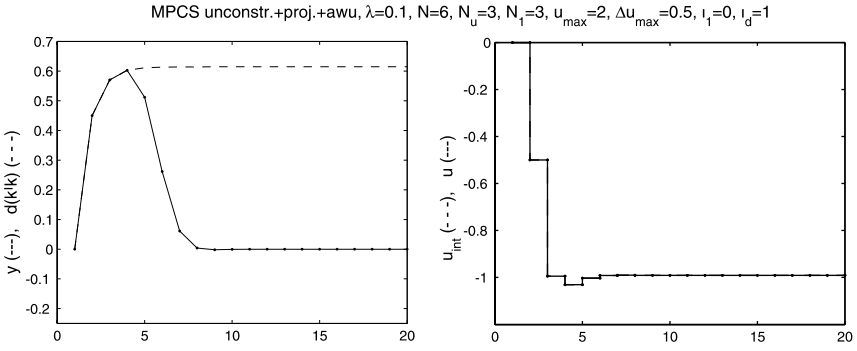


**Fig. 3.37.** Response to a disturbance step in the control system with the explicit MPCS controller, with state estimation, but without modeling errors

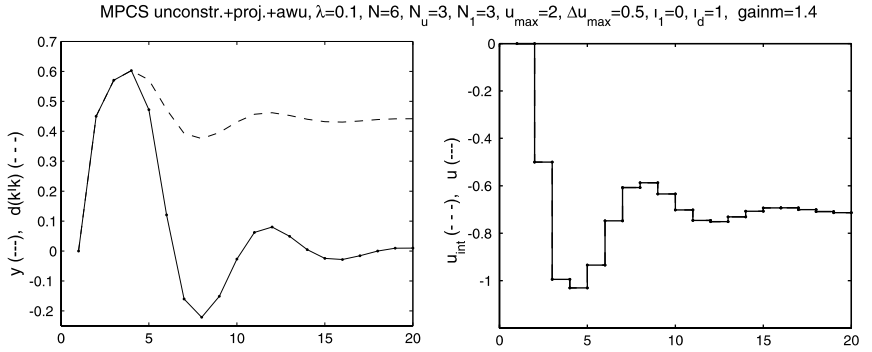


**Fig. 3.38.** Response to a disturbance step in the control system with the explicit MPCS controller, with the gain in the process increased by 40%

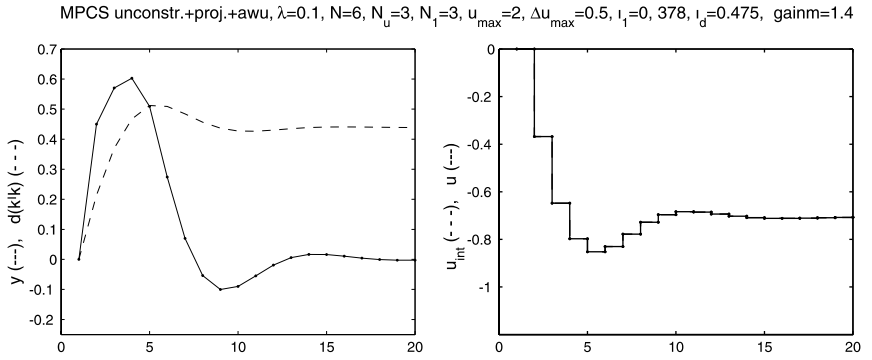


**Fig. 3.39.** Response to a disturbance step in the control system with the explicit MPCS controller, with the process gain increased by 40% and with damped dynamics of the state observer

$\alpha_1 = 0.13$ and $\alpha_5 = 0$, obtaining trajectories almost identical to those presented in Fig. 3.37, only minimally faster – therefore, they will not be given.

Fig. 3.38 presents trajectories with the process gain increased by 40%, but the model unchanged, including the observer parameters. The control system remains stable, but the response becomes oscillatory, and a longer time period is needed for the disturbance estimate to reach the steady-state. To damp oscillations of the disturbance estimate, dynamics of the state observer was changed by taking $\alpha_1 = 0.13$, $\alpha_5 = 0.6$, which results in values $l_1 = 0.378$, $l_d = 0.475$. The resulting trajectories are presented in Fig. 3.39. Finally, in Fig. 3.40 results for the process with the gain increased by 40% and, additionally, an unmodeled dynamics added are presented, with the same observer parameters and, of course, the model unchanged. The process with the unmodeled dynamics was described by the following difference equation

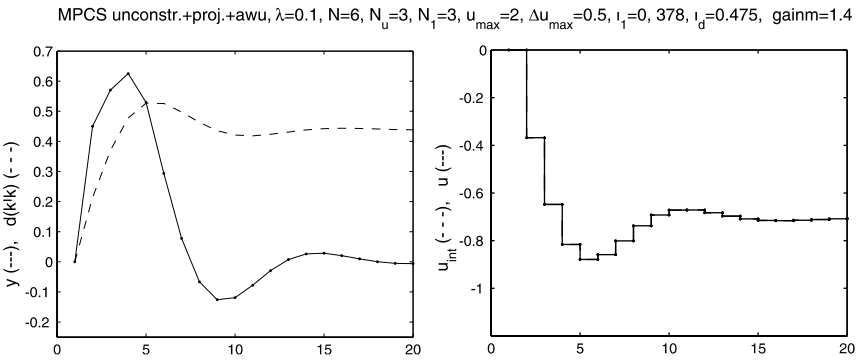


**Fig. 3.40.** Response to a disturbance step in the control system with the explicit MPCS controller, with the process gain increased by 40%, unmodeled dynamics added and damped state observer dynamics

$$(1-0.2676z^{-1}+0.05z^{-2}-0.03z^{-3})y(k) = 1.4(0.1989z^{-2}+0.2552z^{-3})u(k-1)$$

compare with (3.135). □

In this section, devoted to MPC with linear state-space process models, the cases with measured state with additive state disturbances and with estimated state with additive output disturbances were mainly considered. Although more general disturbance models are possible and were mentioned as well, attention was focused on additive disturbances modeled as zero-mean integrated white noises, as of prime importance for industrial applications, delivering zero steady-state errors. Our attention was devoted mainly to prediction equations, disturbance modeling and observer design. Having that, the MPC controller optimization problem is fully defined and can be solved, as an unconstrained or a constrained problem. The structures of the unconstrained MPCS controllers (control laws) have just been derived. A general formulation of the *numerical MPCS constrained optimization problem* is analogous to those given in the previous sections devoted to DMC and GPC algorithms, namely

$$\begin{aligned} \min_{\Delta\mathcal{U}(k)} \{ & \left\| [\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] - \widetilde{\mathbf{C}}\mathbf{M}_x \, \Delta\mathcal{U}(k) \right\|^2_{\underline{\mathbf{\Psi}}} + \|\Delta\mathcal{U}(k)\|^2_{\underline{\mathbf{\Lambda}}} \} \\ \text{subj. to: } & -\Delta\mathcal{U}_{\max} \le \Delta\mathcal{U}(k) \le \Delta\mathcal{U}_{\max} \\ & \mathcal{U}_{\min} \le \mathcal{U}(k-1) + \mathbf{J}\Delta\mathcal{U}(k) \le \mathcal{U}_{\max} \\ & \mathcal{Y}_{\min} \le \mathcal{Y}^0(k) + \widetilde{\mathbf{C}}\mathbf{M}_x \, \Delta\mathcal{U}(k) \le \mathcal{Y}_{\max} \end{aligned} \tag{3.150}$$

where $\mathcal{Y}^0(k)$ is given as in (3.123) or by (3.133) or as in (3.148).

This optimization problem can easily be transformed to a standard form of a QP problem, as it was discussed previously in Section 3.2.4.

When comparing the MPCS with the DMC and GPC algorithms, first of all a different approach to unmeasured disturbances draws our attention. In the DMC and GPC algorithms certain assumptions are made concerning the disturbance models, and under those assumptions formulae were derived for the output prediction. In the DMC algorithm the assumption of a constant output disturbance estimate on a prediction horizon is used, which corresponds to zero-mean integrated white noise added to the output. In the GPC algorithm the model of disturbances can also be in the form of integrated colored noises, although parameters of the coloring filters are usually used as additional tuning parameters. The assumed models are simple and correspond to typical situations in industrial control, and both DMC and GPC algorithms ensure zero control errors in steady-states.

In the MPCS algorithm the model of the process state dynamics can be extended by a model of disturbance dynamics, thus a wider class of disturbances can be modeled – but it is often necessary to design a state observer (a combined one, for states of the process and of disturbances). In the book [82], devoted almost exclusively to predictive control algorithms with state-space models, one can find design examples of MPCS algorithms with different kinds of disturbance models, *e.g.*, periodical.

The necessity to design a state observer can be treated as a drawback, but, on the other hand, it gives additional possibilities in the design of the controller dynamics, as it was shown in Example 3.6. Modeling the dynamics by a set of state-space equations is natural *e.g.*, in the case of many electromechanical objects. But for typical complex industrial processes, where models for control purposes are constructed mainly with the use of input-output type structures, it is rather not a typical solution. In the MPCS algorithm, calculations leading to predicted values of the outputs in a prediction horizon are performed, consequently, on the basis of the state equations. This leads to computationally complex formulae with quite disadvantageous numerical properties, especially when the dimension of the state vector is significantly higher than that of the vector of the controlled outputs and when the prediction horizon is long. This may be especially dangerous for unstable plants. A possible remedy is to pre-stabilize the plant using *e.g.*, a linear state-space feedback, and to design the MPC algorithm for the pre-stabilized system, see *e.g.*, [124, 123].

### 3.4.3 Explicit Piecewise-affine MPCS Constrained Controller

Let us start now with a slightly different form of the the quadratic optimization problem (3.150), putting into it the detailed form of the free output prediction trajectory. The simplest case with measured state and without disturbances will also be assumed, to make the presentation of the main idea of the *explicit constrained* MPC controller with the linear state-space process model more clear. The optimization problem is then in the following form

$$\min_{\Delta\mathcal{U}(k)} \{ \left\| \widetilde{\mathbf{I}}_y y^{sp}(k) - \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}x(k) - \widetilde{\mathbf{C}}\mathbf{V}\mathbf{B}u(k-1) - \widetilde{\mathbf{C}}\mathbf{M}_x \Delta\mathcal{U}(k) \right\|^2_{\underline{\mathbf{\Psi}}} + \|\Delta\mathcal{U}(k)\|^2_{\underline{\mathbf{\Lambda}}} \}$$

$$\begin{aligned} \text{subj. to: } & -\Delta\mathcal{U}(k) \le \Delta\mathcal{U}_{\max} \\ & \Delta\mathcal{U}(k) \le \Delta\mathcal{U}_{\max} \\ & -\mathbf{J}\Delta\mathcal{U}(k) \le -\mathcal{U}_{\min} + \widetilde{\mathbf{I}}_u u(k-1) \\ & \mathbf{J}\Delta\mathcal{U}(k) \le \mathcal{U}_{\max} - \widetilde{\mathbf{I}}_u u(k-1) \\ & \widetilde{\mathbf{C}}\mathbf{M}_x \Delta\mathcal{U}(k) \le -\mathcal{Y}_{\min} + \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}x(k) + \widetilde{\mathbf{C}}\mathbf{V}\mathbf{B}u(k-1) \\ & \widetilde{\mathbf{C}}\mathbf{M}_x \Delta\mathcal{U}(k) \le \mathcal{Y}_{\max} - \widetilde{\mathbf{C}}\widetilde{\mathbf{A}}x(k) - \widetilde{\mathbf{C}}\mathbf{V}\mathbf{B}u(k-1) \end{aligned} \tag{3.151}$$

where $\widetilde{\mathbf{I}}_u = [\mathbf{I}_u\ \mathbf{I}_u\ \cdots \mathbf{I}_u]^T$, $\dim \widetilde{\mathbf{I}}_u = N_u n_u \times n_u$. The problem (3.151) can be reformulated to the following equivalent form

$$\begin{aligned} \min_{\Delta\mathcal{U}(k)} \{\, J(k) &= \frac{1}{2}\Delta\mathcal{U}(k)^T\mathbf{H}\Delta\mathcal{U}(k) + [x(k)^T\, u(k-1)^T\, y^{sp}(k)^T]\mathbf{F}\Delta\mathcal{U}(k) \,\} \\ \text{subj. to: } & \mathbf{G}\Delta\mathcal{U}(k) \le w + \mathbf{E}\begin{bmatrix} x(k) \\ u(k-1) \\ y^{sp}(k) \end{bmatrix} \end{aligned} \tag{3.152}$$

where entries of matrices $\mathbf{H}$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{E}$ and of the vector $w$ depend on entries of matrices and vectors in (3.151), and $[x(k)^T\, u(k-1)^T\, y^{sp}(k)^T]$ is treated as a vector of *parameters*.

If a characterization of the solution to the above optimization problem for a full range of its parameter values is sought, *i.e.*, as a function of these parameters, than the problem is qualified as a *multi-parametric quadratic programming* (mp-QP) problem. The mp-QP approach to the MPCS design was thoroughly investigated in [7], a short description in this section follows the idea presented there.

First, changing the vector of the decision variables,

$$z(k) = \Delta\mathcal{U}(k) + \mathbf{H}^{-1}\mathbf{F}^T[x(k)^T\, u(k-1)^T\, y^{sp}(k)^T]^T \tag{3.153}$$

the problem (3.152) is further reformulated to a simpler form, with the parameter vector appearing only on the right hand side of the constraints,

$$\begin{aligned} \min_{z(k)}\{\, J_z(k) &= \frac{1}{2}z(k)^T\mathbf{H}z(k) \,\} \\ \text{subj. to: } & \mathbf{G}z(k) \le w + \mathbf{W}\begin{bmatrix} x(k) \\ u(k-1) \\ y^{sp}(k) \end{bmatrix} \end{aligned} \tag{3.154}$$

where

$$\mathbf{W} = \mathbf{E} + \mathbf{G}\mathbf{H}^{-1}\mathbf{F}^T$$

and the difference between both objective functions is in a parameter-dependent term only,

$$J_z(k) = J(k) + \frac{1}{2}[x(k)^T\, u(k-1)^T\, y^{sp}(k)^T]\mathbf{F}\mathbf{H}^{-1}\mathbf{F}^T[x(k)^T\, u(k-1)^T\, y^{sp}(k)^T]^T$$

In [7] results of a thorough theoretical analysis of the mp-QP problems of type (3.154) are presented. It is proved that for every set of *the same active inequality constraints* the solution to (3.154) is an unique affine function of the parameters, *i.e.*, as long as activity of the constraints does not change, *the constrained MPCS control law is an affine function of the parameter vector*. This result could have been awaited, as resulting from the structure of the problem, and was previously known, *e.g.*, [157]. However, to apply it in practice, a computable characterization of all subsets of the parameter space, over which the different affine control laws are defined, must be known. This characterization is the main result of the paper [7].

It is shown in [7] that the whole parameter space can be divided into polyhedral regions of activity of the same constraints. A representation of these regions is given, based on checking signs of Lagrange multipliers corresponding to the active constraints. Based on these results, an algorithm is proposed for partitioning the parameter space. It will not be given here since it is quite complicated, and the presentation would have to be precise to be useful. Moreover, in [143, 144] improved algorithms for parameter space partitioning were proposed, leading to smaller numbers of polyhedral regions. The interested reader is referred to the cited literature and references therein.

The main advantage of the explicit piecewise-affine MPCS algorithm is that an on-line numerical solution of the QP problem is avoided, thus the algorithm can potentially be applied to constrained processes with faster dynamics, without any loss in optimality. The main computational burden is shifted to off-line calculations, to the design phase.

Certainly, the algorithm is more complicated than a suboptimal explicit unconstrained control law with a projection onto the constraint set and anti-windup scheme added, because the collection of all affine control laws together with descriptions of corresponding polyhedral regions must be stored, and a selection mechanism must be applied on-line – but the algorithm is optimal.

The discussion shows that a critical factor for applicability of the explicit piecewise-affine MPCS controller is the possibly large number of polyhedral regions dividing the parameter space. This number depends directly on the number of inequality constraints in (3.151). This, in turn, depends on dimensionality of the input vector $u(k)$ and on the length of the control horizon $N_u$. Further, if constraints on the output vector $y(k)$ are present, their number depends on dimensionality of $y(k)$ and on the length of the prediction horizon $N$, if the outputs are constrained at every sampling instant over this horizon. To diminish the number of the output constraints, the use of a shorter *constraint horizon* $N_c < N$ is proposed to be used if possible, or a narrower *constraint window* $[N_{cw1},\, N_{cw}]$, $N_1 \leq N_{cw1} < N_{cw} \leq N$, as defined in Section 3.1.

Notice also that the simplest case, with measured state and without disturbances, has been considered so far. Both measured and unmeasured disturbances must be added to the parameter vector (usually estimated together with the state vector by an observer, as discussed earlier in the proceeding sections), thus enlarging dimensionality of the parameter space of the mp-QP problem. But, as claimed in [7], this should not enlarge significantly the number of polyhedral regions, because adding disturbances do not introduce new explicit inequality constraints.

Undoubtedly, the explicit MPCS controller is an appealing alternative, especially for processes with fast dynamics and when much simpler suboptimal unconstrained explicit control laws supplemented by projections onto constraint sets and anti-windup schemes cannot be accepted, especially in cases with constraints on process outputs. On the other hand, the still increasing power of industrial controllers makes applications of numerical MPC algorithms, with an on-line QP optimization, possible to still wider classes of plants. As well, the design and tuning of these algorithms is much simpler, especially when input-output type models are available. Moreover, in cases of a large number of regions finding the right one may take more time than solving a QP problem. Last but not least, there is always uncertainty in process modeling, especially in process control. When model-reality differences are significant, then the explicit piecewise-affine control law will also be approximate, suboptimal. Analysis and comparison of different MPC algorithms in this situation is an appealing question.

## 3.5 Nonlinear Predictive Control Algorithms

### 3.5.1 Structures of Nonlinear MPC Algorithms

Numerous industrial applications of predictive control algorithms with linear process models were undeniably a milestone in the development of theory and practice of feedback control. Real control processes are, however, usually nonlinear and a linear feedback controller designed for a vicinity of the assumed operating point is not always sufficient. Especially, for optimal control in a multilayer structure, when the optimizer changes operating points on-line to adjust them to current values of uncontrolled inputs, there is often a need for a nonlinear control, see Chapters 1 and 4. Another case often calling for a nonlinear approach is the control of batch processes, where dynamic set-point trajectories are typical. Thus, already in the 1980s attempts to formulate MPC algorithms for nonlinear process models appeared, especially within the community connected with development and applications of the DMC algorithm in chemical industries. The dominant approach was that using successive linearizations, started by Garcia [47], see [8, 55]. The decade of the 1990s was a period of growing research on nonlinear control algorithms, see the review articles [55, 1, 94, 115].

Applications of MPC algorithms with a full nonlinear model used in the controller optimization problem, solved at every sampling instant, have so far found a limited application, but are the important subject of a research. Such algorithms can be applied mainly for slow but strongly nonlinear processes, when aggressive control can be expected. Basic problems connected with the design and implementation of algorithms with a nonlinear optimization at each sampling instant will be described briefly in the first part of this section.

A clearly visible tendency in publications connected with practical applications, is to apply *MPC algorithms with nonlinear model linearizations*, if there is a need for nonlinear modeling and control. M.Morari and J.H.Lee wrote in a review paper [98] that *linearization is the only mehod which found a wider application in industry, except for demonstration projects. For industry there must be a clearly justified need for on-line solving of nonlinear dynamic tasks, and so far there have been no examples indicating such a need in a convincing way.*

Using the linearization we usually aim at a formulation of the MPC algorithm such that at each sampling instant a convex quadratic programming problem is appropriately constructed and solved, once or even a few times in a computational loop, in this way adapting the algorithm to the model nonlinearity. The following algorithmic solutions using a general nonlinear model in a MPC structure by its appropriate linearization at each sampling instant should be quoted here, see [79, 50, 77, 101, 95, 137], in the review paper [55] one can also find a more extensive bibliography.

Many publications appeared in the last decade concerning stability analysis of many different predictive control algorithms using a general nonlinear model. However, it should be noted that there are very few such publications in the area of algorithms with linearizations. This view is also shared by M. Morari and J.H. Lee, who wrote in the mentioned review paper that *theoretical purists tend to stay away from linearization approaches.* Certainly, one of the basic reasons for this situation is the fact that algorithms with linearizations are *suboptimal* – and algorithms of this type are much more difficult for a formal theoretical analysis. However, from the very beginning of the era of the MPC algorithms, a lack of theoretical analysis has not been and still is not a decisive factor in their development and application.

Structures of nonlinear, suboptimal MPC algorithms using appropriate linearizations of a nonlinear process model will be the main subject of consideration in this section.

### 3.5.2 MPC-NO (MPC with Nonlinear Optimization)

At a first glance, *the idea how to generalize a linear to a nonlinear predictive control* seems fairly straightforward: leaving the design unchanged, one should only use a *nonlinear, instead of a linear*, process model for predicting the outputs over a prediction horizon. This leads to the formulation of the controller optimization problem in the following form (compare with (3.11))

$$\min_{\triangle u(k|k),\ldots,\triangle u(k+N_u-1|k)} \{ \sum_{p=N_1}^{N} \|y^{sp}(k+p|k) - y(k+p|k)\|^2_{\mathbf{\Psi}(p)} + \\ + \sum_{p=0}^{N_u-1} \|\triangle u(k+p|k)\|^2_{\mathbf{\Lambda}(p)} \}$$

subj. to: (3.7), (3.8) and (3.9), where

the dependence of $y(k+p|k)$ on past process outputs and inputs and on decision variables $\triangle u(k+p|k)$, $p = 0, ..., N_u - 1$, is given by a *nonlinear model.*

(3.155)

We shall describe a full nonlinear MPC algorithm, with the prediction of the entire output trajectory based on a nonlinear model, as the *MPC-NO (MPC with Nonlinear Optimization)* algorithm. The structure of this algorithm is illustrated in Fig. 3.41, where $\mathcal{Y}^{sp}(k) = [y^{sp}(k+N_1|k)^T \ \cdots \ y^{sp}(k+N|k)^T]^T$ represents a trajectory of the set-points.

However, a seemingly small difference when a nonlinear process model is used instead of a linear one, is fundamental. A nonlinear dependence of predicted outputs $y(k+p|k)$ on decision variables causes the fact that the optimization problem (3.155) becomes *non-quadratic and, generally, non-convex.* There are no universal optimization procedures for solving such problems which would ensure finding a solution reliably and quickly, *i.e.*, with a guaranteed accuracy and within an a priori prescribed time interval. Moreover, more effective gradient optimization procedures usually find only local minima. Therefore, in spite of many research results concerning the subject, applications of the MPC algorithms with a nonlinear model in the optimization problem, which is solved at every sampling instant by a selected procedure of

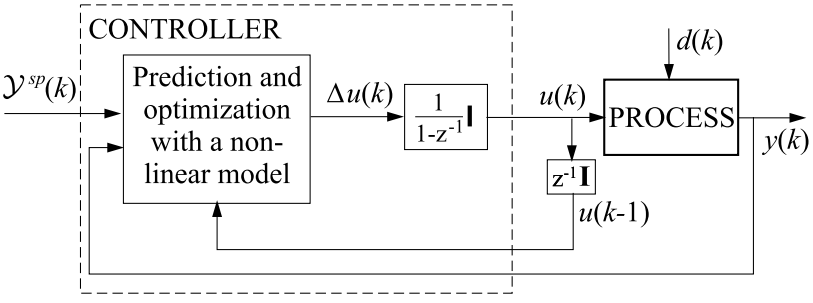


**Fig. 3.41.** Structure of full nonlinear model predictive control, with both prediction and optimization based on a nonlinear model – a MPC-NO structure

nonlinear programming, found so far very limited application. They are designed mainly for strongly nonlinear processes with slower dynamics. Henson, in his review article [55], gives a list of examples of simulations of predictive control algorithms with nonlinear models, and a much shorter list of true experimental applications.

It is worth noting that one of the promising research directions in the area of nonlinear predictive control is the application of neural models with certain structures, see *e.g.*, [108, 72, 71, 73, 67, 136]. This topic will be addressed later on in this chapter. Another interesting approach is an attempt to use a branch-and-bound method for a nonlinear, global optimization of a suitable modified optimization problem, at each sampling instant of the MPC-NO algorithm [128]. To apply the branch-and-bound method it was necessary to discretize each of the decision variables $\triangle u(k+p|k)$, $p = 0, 1, ..., N_u - 1$. As a result, a suboptimal algorithm was obtained, with the level of suboptimality as well as the computational burden depending on the accuracy of this discretization. In order to obtain an acceptable amount of calculations, in [147] discretization was restricted to only selected subsets of possible control input values, located around the previously calculated ones, additionally scaled adaptively with the use of fuzzy logic. This approach, though claimed as successful, turned out to require still too high amount of calculations.

Finally, let us emphasize that most attention of the control theorists is devoted exactly to the MPC-NO algorithms. A number of different structures of such algorithms was proposed and still there appear concepts of this type, often together with a stability analysis. We shall go back to this subject in Section 3.6.1, devoted to stability analysis of predictive control algorithms.

### 3.5.3 MPC-NSL (MPC Nonlinear with Successive Linearization)

When considering linear MPC algorithms, it was emphasized that differences between them result from using different process models for prediction of the output values in a prediction horizon. This led to various structures and forms of formulae for the free component of the predicted outputs trajectories and only different representations of the same formula for the forced component, defined by the dynamic matrix. In the case of a *nonlinear process model*, nonlinear equations define the predicted trajectory of the controlled outputs $y(k+p|k)$, $p = 1, 2, ..., N$. Because then the principle of superposition does not hold, it is generally impossible to decompose this trajectory, without a loss of optimality, into nonlinear forced and free components: the forced one dependent on decision variables (process input increments to be found) and the free one dependent on past process input and output values. However, inserting the process input increments equal to zero over the prediction horizon into nonlinear equations describing the process input-output model, we transform these equations into a *nonlinear model of a free trajectory* of the predicted outputs, $y^0(k+p|k)$, $p = 1, 2, ..., N$. This way of proceeding will not, however, be constructive if we are not able to derive independent and simple formulae

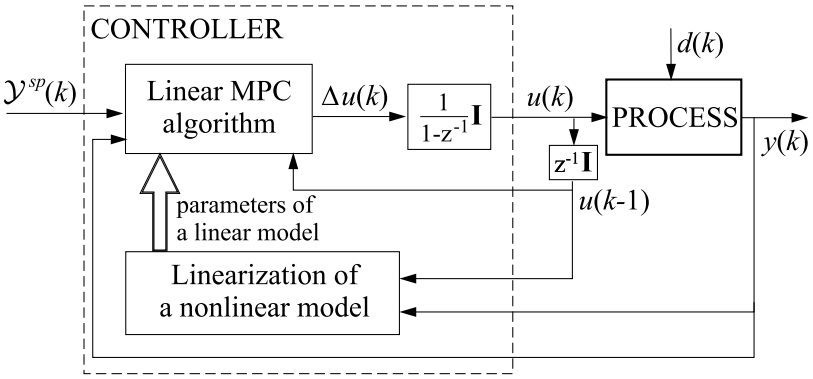


**Fig. 3.42.** Structure of a nonlinear MPC algorithm with successive linearizations of the process model (MPC-NSL algorithm)

for a forced component of the predicted outputs trajectory. The reason is that, when minimizing the cost function in (3.155), it will still be necessary to use full nonlinear equations to calculate the values $y(k+p|k)$, $p = 1, 2, ..., N$, as dependent on the decision variables $\triangle u(k+p|k)$, $p = 0, ..., N_u - 1$, after each change of values of these variables by a numerical optimization procedure.

The simplest solution which enables a natural generalization of well-known good practical properties of linear predictive algorithms (*i.e.*, with linear process models) to nonlinear predictive control, are algorithms which at each sampling instant perform a *linearization of the nonlinear model*, at a current process state, and then calculate the control inputs using a linear MPC algorithm with the linearized model, see *e.g.*, [55, 95, 4, 89, 137, 75]. This is a *suboptimal approach*, but the one which enables to keep a fundamental, for practical applications, feature of a *control reliability*, *i.e.*, a guarantee that at each of the successive algorithm steps an optimal solution of the (quadratic) optimization problem will be found, and always within a predefined time. The discussed structure of a suboptimal nonlinear MPC control with model linearizations will be called *MPC-NSL* (*MPC Nonlinear with Successive Linearizations*). It is presented in Fig. 3.42.

For weakly nonlinear processes, or operating close to certain equilibrium points during longer time periods and under slowly-varying disturbances, the linearization may not be necessary at each sampling instant. It may then be sufficient to perform it more rarely, *e.g.*, only after a defined number of samples. Such an example is given in [82]. On the other hand, the presented MPC-NSL algorithm may occur to be not sufficiently nonlinear for stronger nonlinearities, for situations when a quick transfer to a distant set-point is needed or for dynamic transients after strong and rapid changes of disturbances. In these cases it may be reasonable to apply MPC algorithms which more

directly base on the nonlinear model. This leads, first of all, to algorithms with a nonlinear prediction and linearization, which will be presented in the next section.

### 3.5.4 MPC-NPL (MPC with Nonlinear Prediction and Linearization)

A structurally more precise algorithmic solution than the MPC-NSL algorithm presented above, and at the same time still simple in implementation and preserving the required good properties of a quadratic programming optimization problem, is to use a linearized model only in the optimization problem, *i.e.*, with only a forced trajectory of controlled outputs linearly depending on decision variables, but evaluating the free trajectory of predicted outputs from a nonlinear model, at each sampling instant, see [47, 50, 55, 95, 147, 89, 137, 73]. This structure will be called *MPC-NPL* (*MPC with Nonlinear Prediction and Linearization*).

In fact, *there is no reason* to give up a nonlinear prediction of a free output trajectory having a nonlinear process model. This is a relatively easy task, performed only once at each sampling instant of the MPC controller. Thus, it does not significantly affect the total amount of calculations performed at each sampling instant, which is first of all determined by a numerical solution of the optimization problem, even if it is a quadratic programming problem. Thus, the MPC-NPL algorithms should be preferred to the MPC-NSL algorithms. Of course, not in cases when our knowledge about the problem or results of simulation tests show that it is quite sufficient to use an MPC-NSL algorithm. Particularly, in a situation when it is enough to perform the linearization only once every several sampling instants.

To calculate the free component $\mathcal{Y}^0(k)$ of a trajectory of predicted outputs on a prediction horizon it is necessary to have not only values of the already applied (past) process input and output signals, but also to assume certain (initial) values of *future process inputs* in the prediction horizon. We shall denote the trajectory of these future process input values by $\mathcal{U}^0(k)$. Proceeding identically as in the case of MPC algorithms with linear models, in order to evaluate the free output trajectory we can assume zero increments of process input signals in a prediction horizon, *i.e.*, process inputs constant and equal to $u(k-1)$,

$$\mathcal{U}^0(k) = \begin{bmatrix} u^0(k|k) \\ u^0(k+1|k) \\ \vdots \\ u^0(k+N-1|k) \end{bmatrix} = \begin{bmatrix} u(k-1) \\ u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix} \tag{3.156}$$

This is not the only possible approach. It is also possible to define $\mathcal{U}^0(k)$ using an optimal control trajectory evaluated at the previous algorithm step, in the following way

$$\mathcal{U}^0(k) = \begin{bmatrix} u^0(k|k) \\ u^0(k+1|k) \\ \vdots \\ u^0(k+N-3|k) \\ u^0(k+N-2|k) \\ u^0(k+N-1|k) \end{bmatrix} = \begin{bmatrix} \hat{u}(k|k-1) \\ \hat{u}(k+1|k-1) \\ \vdots \\ \hat{u}(k+N-3|k-1) \\ \hat{u}(k+N-2|k-1) \\ \hat{u}(k+N-2|k-1) \end{bmatrix} \tag{3.157}$$

Note that in the case of a linear process model, using (3.157) in place of (3.156) leads to the same solution of the optimization problem, due to linearity of both free and forced output responses. Therefore, a simpler formulation (3.156) is used. However, it is not equivalent when a nonlinear process model is used, application of (3.157) may then lead to different results, although the difference is usually not very significant [87].

In an MPC-NPL algorithm elements of the predicted trajectory of the controlled variables at sampling instant $k$ are evaluated as follows

$$\begin{aligned} y(k+p|k) &= y^0(k+p|k) + \triangle y(k+p|k) \\ &= y^0(k+p|k) + \overline{\mathbf{M}}_p(k)\Delta\mathcal{U}(k), \quad p = N_1, ..., N \end{aligned}$$

where $y^0(k+p|k)$ are elements of the free output trajectory $\mathcal{Y}^0(k)$, while the forced trajectory $\Delta\mathcal{Y}(k)$ is calculated as follows

$$\begin{aligned} \Delta\mathcal{Y}(k) &= \begin{bmatrix} \triangle y(k+N_1|k) \\ \vdots \\ \triangle y(k+N|k) \end{bmatrix} \\ &= \mathbf{M}(k)\triangle\mathcal{U}(k) = \begin{bmatrix} \overline{\mathbf{M}}_{N_1}(k) \\ \vdots \\ \overline{\mathbf{M}}_N(k) \end{bmatrix} \begin{bmatrix} \triangle u(k|k) \\ \vdots \\ \triangle u(k+N_u-1|k) \end{bmatrix} \end{aligned} \tag{3.158}$$

Thus, the quadratic programming problem will have the following form

$$\begin{aligned} &\min_{\Delta\mathcal{U}(k)} \{ \left\| [\mathcal{Y}^{sp}(k) - \mathcal{Y}^0(k)] - \mathbf{M}(k)\Delta\mathcal{U}(k) \right\|^2_{\underline{\boldsymbol{\Psi}}} + \|\Delta\mathcal{U}(k)\|^2_{\underline{\boldsymbol{\Lambda}}} \} \\ &\text{subj. to:} \ -\Delta\mathcal{U}_{\max} \le \Delta\mathcal{U}(k) \le \Delta\mathcal{U}_{\max} \\ &\qquad \mathcal{U}_{\min} \le \mathcal{U}^0(k) + \mathbf{J}\Delta\mathcal{U}(k) \le \mathcal{U}_{\max} \\ &\qquad \mathcal{Y}_{\min} \le \mathcal{Y}^0(k) + \mathbf{M}(k)\Delta\mathcal{U}(k) \le \mathcal{Y}_{\max} \end{aligned} \tag{3.159}$$

compare with (3.69) and (3.155). In this problem, the *input increments* $\Delta\mathcal{U}(k)$, and thus the output increments $\Delta\mathcal{Y}(k)$, are evaluated *with reference to the nonlinear trajectory* $\mathcal{Y}^0(k)$, while $\mathbf{M}(k)$ denotes a matrix of a linear mapping evaluated as a current, local linear approximation (linearization) of the nonlinear model. Of course, elements of this matrix are coefficients of the

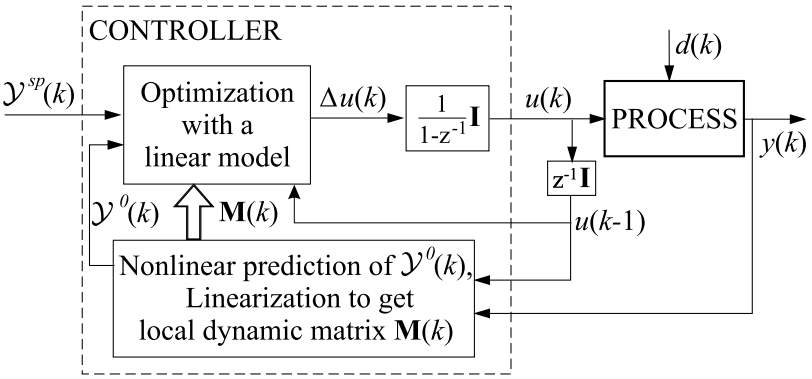


**Fig. 3.43.** Structure of the MPC algorithm with nonlinear prediction of the free output trajectory and current nonlinear model linearization to get linear forced output trajectory (MPC-NPL algorithm)

step response of the linearized model. The structure of the *MPC-NPL control algorithm* is presented in Fig. 3.43.

We have already explained that the smaller control input increments evaluated as a solution of the quadratic optimization problem, the better the MPC-NPL algorithm should operate. Therefore, it may deliver almost optimal behavior even for strongly nonlinear processes and for transitions to distant operating points, as long as the trajectories are smooth and realized with small process input changes. If, however, fast and large changes in operating points are required, or a fast reaction to step disturbances of large amplitudes is required, then using even the MPC-NPL algorithm may not be satisfactory. In the literature, it is possible to find suggestions on how to improve features of nonlinear MPC algorithms still maintaining a linear-quadratic form of the current optimization problem and at a cost of a limited increase of computational burden. The main indication is to apply iterative, nonlinear corrections of the free response trajectory and of the quadratic programming problem, at every controller step (sampling instant) [79, 77, 101, 55, 30].

A conceptually clear and relatively easy method of improving the nonlinear features of the MPC-NPL algorithm when larger changes of the control input signal occur is to repeat, once or a few times, a computational loop consisting of the evaluation of a free output trajectory over the prediction horizon and then a resulting quadratic programming problem. This method is not beyond present levels of the computing power of modern industrial processors. Repeating a numerical solution of a small – or even moderate-size – quadratic programming problem in a small fraction of one sampling period should not be a problem, especially for processes with slower dynamics.

The most important is to improve nonlinear features of the MPC-NPL algorithm at the first sampling instant of the control horizon, where the first

control input increment $\triangle\hat{u}(k|k)$ is calculated. There are two reasons for this: first of all, after a larger change of a set-point or a disturbance, the algorithm reacts stronger usually at this first moment, *i.e.*, choosing then the largest change of the control input. Secondly, only the first element of a control input trajectory becomes the actual process control input signal, the following ones will be calculated anew according to the general rule of the receding horizon.

Therefore, a relatively simple, yet effective way of improving nonlinear features of the MPC-NPL algorithm can be proposed as iterative corrections of the nonlinear free output response and of the resulting quadratic programming problem. This will be done by means of an iterative loop modifying the optimized process input and output values at sampling instant $k$. From now on, we shall refer to such an extension of the MPC-NPL algorithm as to the *MPC-NPL+ algorithm.*

The *structure of the MPC-NPL+ algorithm* calculations, performed for one sampling interval, is as follows:

1. Using a nonlinear model, on the basis of previous process output and input values and an initial trajectory of the process control inputs $\mathcal{U}^0(k)$ (3.156) calculate the free trajectory of the predicted outputs $\mathcal{Y}^0(k)$. Calculate the dynamic matrix of the linearized model $\mathbf{M}(k)$. Set the index of internal iterations $j = 0$, set $\Delta\mathcal{U}^0(k) = 0$.
2. Solve the quadratic programming problem

$$\begin{aligned} \min_{\Delta\mathcal{U}(k)} \{ \left\| [\mathcal{Y}^{sp}(k) - \mathcal{Y}^j(k)] - \mathbf{M}(k)\Delta\mathcal{U}(k) \right\|^2_{\underline{\mathbf{\Psi}}} + \left\| \Delta\mathcal{U}^j(k) + \Delta\mathcal{U}(k) \right\|^2_{\underline{\mathbf{\Lambda}}} \} \\ \text{subj. to: } -\Delta\mathcal{U}_{\max} \le \Delta\mathcal{U}^j(k) + \Delta\mathcal{U}(k) \le \Delta\mathcal{U}_{\max} \\ \mathcal{U}_{\min} \le \mathcal{U}^j(k) + \mathbf{J}\Delta\mathcal{U}(k) \le \mathcal{U}_{\max} \\ \mathcal{Y}_{\min} \le \mathcal{Y}^j(k) + \mathbf{M}(k)\Delta\mathcal{U}(k) \le \mathcal{Y}_{\max} \end{aligned} \tag{3.160}$$

   obtaining optimal increments $\triangle\hat{u}^{j+1}(k+p|k)$, $p = 0, ..., N_u - 1$.
3. If

$$\|\triangle\hat{u}^{j+1}(k|k) - \triangle\hat{u}^j(k|k)\| < \varepsilon\|\triangle\hat{u}^j(k|k)\|$$

   (where $\triangle\hat{u}^0(k|k) = u^0(k|k) - u(k-1) = 0$, see (3.156)) then *terminate* and transmit the final first control input increment, $\triangle\hat{u}(k)$,

$$\triangle\hat{u}(k) = \triangle\hat{u}^{j+1}(k|k) + u^j(k|k) - u(k-1)$$

   to the controlled process. Otherwise go to Step 4.
4. Perform a *correction of the free output trajectory*: using the nonlinear model calculate the trajectory of predicted outputs $\mathcal{Y}^{j+1}(k)$, consisting of elements $y^{j+1}(k+p|k)$, $p = 1, ..., N$, calculated for the past outputs and inputs and future inputs computed for the prediction horizon and corrected by increments $\triangle\hat{u}^j(k|k)$ calculated in Step 2. Thus, the trajectory $\mathcal{Y}^{j+1}(k)$ is calculated for the following vector of process inputs

$$\mathcal{U}^{j+1}(k) = \begin{bmatrix} u^{j+1}(k|k) \\ u^{j+1}(k+1|k) \\ \vdots \\ u^{j+1}(k+N-1|k) \end{bmatrix} = \begin{bmatrix} u^{j}(k|k) + \triangle\hat{u}^{j+1}(k|k) \\ u^{j}(k|k) + \triangle\hat{u}^{j+1}(k|k) \\ \vdots \\ u^{j}(k|k) + \triangle\hat{u}^{j+1}(k|k) \end{bmatrix}$$

where $u^0(k|k) = u(k-1)$. Further, calculate

$$\Delta\mathcal{U}^{j+1}(k) = \begin{bmatrix} u^{j+1}(k|k) - u(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

set $j = j + 1$ and go back to Step 2.

Having modified the free output trajectory in Step 4, it is necessary to *correct the quadratic programming problem*, thus in Step 2 a modified problem (3.160) is solved, which reduces to (3.159) for $j = 0$ (first optimization). Control input increments are now calculated in the optimization problem with reference to the modified free output trajectory. At the same time, the penalty terms for the input increments in the cost function and in the constraints must contain the total input increments, calculated with reference to the last applied process input $u(k-1)$, thus the occurrence of vectors $\Delta\mathcal{U}^j(k)$ and $\mathcal{U}^j(k)$ in the cost function and in the constraints.

The value of the coefficient $\varepsilon$ occurring in the termination criterion for internal iterations in Step 3 of the algorithm should not be too small, because only for a significant difference a repetition of prediction and optimization should take place. Theoretically, iterating process can require several steps. However, due to the necessity to finish all computations needed to evaluate the final control increment in a predefined time period, the maximum number of iterations is usually additionally arbitrarily limited to a very few repetitions. The more so, because the convergence of this type of iterations (though usually fast convergent) is generally not guaranteed. Thus, when allowing multiple iterations, a protection mechanism should be implemented which interrupts iterations if necessary.

The structure of the MPC-NPL+ algorithm is presented in Fig. 3.44. Note that its iterative loop can also be treated as *built-in extension of the MPC-NPL algorithm*, activated only at sampling periods when strongly nonlinear behavior of the process must be taken into account.

Another question which needs to be discussed is that of obtaining a linear model approximation at each sampling instant, *i.e.*, the problem of calculation of the dynamic matrix $\mathbf{M}(k)$ which defines linear dependence between process input and output increments, needed in the quadratic programming problem. In the MPC-NSL algorithm, at each sampling instant a linear approximation of the nonlinear model at a current process state is calculated first. Then, for the obtained linear model, the optimal control input increment is calculated

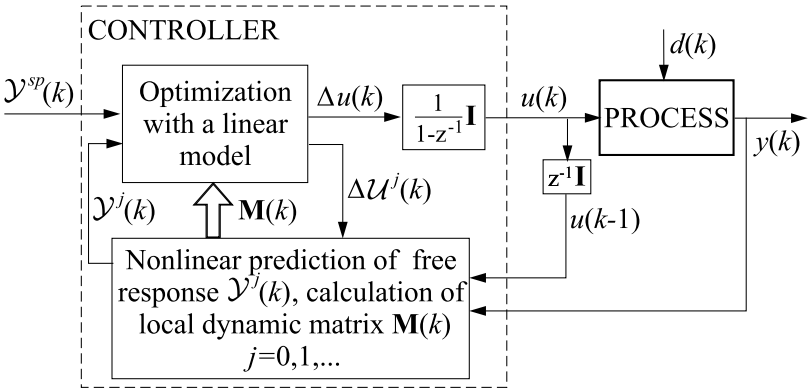


**Fig. 3.44.** Improved MPC-NPL algorithm structure with iterative improvement of the prediction and optimization added (MPC-NPL+ algorithm)

by a selected linear MPC algorithm, usually one of the previously described algorithms: DMC, GPC or MPCS. Thus, the matrix $\mathbf{M}(k)$ is calculated as a dynamic matrix of the applied linear model, using a formula corresponding to the chosen MPC algorithm.

In the case of the MPC-NPL algorithm the situation is not so obvious, as a predicted free output trajectory is calculated in a nonlinear way. Consequently, during its calculation the predicted process state will change when moving from the beginning to the end of the prediction horizon, generally in a nonlinear way. Therefore, a linear process approximation at each of the subsequent points within the prediction horizon can be different, and can be obtained by performing a sequence of linearizations (a linearization around a nonlinear free output trajectory). As a result, the matrix $\mathbf{M}(k)$, see (3.158) will consist of sub-matrices (in cases of a scalar output $y$ – rows) $\overline{\mathbf{M}}_p(k)$ calculated for slightly different linear models corresponding to individual sampling instants of the prediction horizon, $p = 1, ..., N$. In the case of the MPC-NPL+ algorithm considered, the calculation of the dynamic matrix can be additionally repeated after each modification of the free output trajectory in Step 4. However, the calculation of this matrix should then be transferred from Step 1 to the beginning of Step 2 of the MPC-NPL+ algorithm, it will also then be an indexed matrix $\mathbf{M}^j(k)$.

The published results of simulations [147, 89, 87], though fragmentary for obvious reasons, and the author's experience, indicate that the *nonlinear* method presented above for the calculation of the matrix $\mathbf{M}(k)$ does not usually lead to a significant improvement. Thus, it can be considered only in strongly nonlinear cases. Further, since the influence of first steps of the prediction horizon is most important, therefore the described nonlinear modifications can be limited only to an initial fragment of the prediction horizon.

The MPC-NPL+ structure is the most developed one from amongst the nonlinear predictive methods using a linear process model in the optimization problem, it is addressed to strongly nonlinear problems and more demanding situations, when it is necessary to use appropriate control reactions to large and fast changes of set-points or disturbances. Moreover, it is possible to try to make this structure even more elaborated, two rather obvious extensions are possible:

- Instead of a constant initial process input trajectory (3.156), the calculations can be based on the initial trajectory (3.157).
- The vector of future process inputs can be corrected by a full trajectory of input increments calculated in the quadratic programming problem, and not only by its first element $\triangle\hat{u}^{j+1}(k|k)$.

A fairly obvious formulation of the MPC-NPL+ algorithm including one or both extensions given above, is left to the reader.

The presented structures of nonlinear MPC control algorithms: the optimal structure (MPC-NO) and the suboptimal ones (MPC-NSL, MPC-NPL) are fairly general, they do not limit possible forms of nonlinear models. The control problem can be with constraints, at each step of an MPC algorithm a nonlinear optimization problem (in MPC-NO) or a quadratic convex optimization problem (in MPC-NSL and MPC-NPL) is solved, with constraints.

A special and important case, from an application point of view, is that of a *nonlinear process modeled by a TS (Takgi-Sugeno) fuzzy system* with linear functional consequents in the fuzzy rules, see Section 2.1. For such a nonlinear model it is easy to generate local linear models which are needed at consecutive sampling instants of nonlinear MPC algorithms with linearizations. Moreover, in cases when a controller without constraints is designed, it is possible to use a general nonlinear TS fuzzy control structure with output-feedback or state-feedback, see Section 2.2. We define then consequents of the rules of a nonlinear controller as linear, explicit MPC control laws. It is worth remembering that analytical versions of the MPC control laws implemented correctly in anti-wind-up structures are of practical importance. Moreover, when designing the MPC controllers, it is sometimes recommended to design first a controller without constraints, see *e.g.*, [49, 99]. It is then often easier to select suitable controller parameters which influence dynamics and robustness of the control system.

For notation simplicity, we shall define MPC algorithms with nonlinear TS fuzzy process models as *FMPC (Fuzzy Model Predictive Control) algorithms*. The presented so far nonlinear MPC algorithms will be described in more detail for the TS fuzzy models in Sections 3.5.7 and 3.5.8.

**Example 3.7**
Consider an example of a process model dynamics described by a nonlinear function $g : \mathbb{R}^3 \to \mathbb{R}$

$$y(k+1) = g(y(k), u(k-1), u(k-2)) \tag{3.161}$$

assuming also an additive, constant output disturbance model. We shall now demonstrate how to construct free output trajectories and the matrix $\mathbf{M}(k)$ which defines a forced output trajectory in the algorithms with linearizations, MPC-NSL and MPC-NPL. Assume the lengths of control and prediction horizons $N_u = 4$ and $N = 6$. There is a time delay $\tau = 1$ in the process, thus $N_1 = 2$ is assumed.

Define the vector $x(k) \in \mathbb{R}^3$, $x(k) = [y(k)\ u(k-1)\ u(k-2)]^T$. Denote the *linearization point* by $x_L(k)$, $x_L(k) = [y_L(k)\ u_L(k-1)\ u_L(k-2)]^T$, where $y_L(k)$ is a measurement $y(k)$ of the output at sampling instant $k$, whereas $u_L(k-1)$ and $u_L(k-2)$ are process input signals $u(k-1)$ and $u(k-2)$ at previous instants. Moreover, denote

$$a_1(x_L(k)) = -\frac{\partial g}{\partial x_1}(y_L(k), u_L(k-1), u_L(k-2)) \tag{3.162a}$$

$$b_1(x_L(k)) = \frac{\partial g}{\partial x_2}(y_L(k), u_L(k-1), u_L(k-2)) \tag{3.162b}$$

$$b_2(x_L(k)) = \frac{\partial g}{\partial x_3}(y_L(k), u_L(k-1), u_L(k-2)) \tag{3.162c}$$

A linearized form of the dynamics equation at the point $x_L(k)$ is the following

$$\begin{aligned} y(k+1) = {} & g(x_L(k)) - a_1(x_L(k))[y(k) - y_L(k)] + \\ & + b_1(x_L(k))[u(k-1) - u_L(k-1)] + b_2(x_L(k))[u(k-2) - u_L(k-2)] \end{aligned} \tag{3.163}$$

For a concise notation, let us further denote:

$$a_1(k) = a_1(x_L(k)), \quad b_1(k) = b_1(x_L(k)), \quad b_2(k) = b_2(x_L(k))$$

- *MPC-NSL algorithm*

The MPC-NSL algorithm will be a standard formulation of a linear MPC algorithm designed for the linearized model (3.163). For this purpose we can use one of the design methods presented in previous sections, *e.g.*, for the GPC algorithm. For the needs of the considered example we shall use directly the equation (3.163), to calculate recurrently elements $y^0(k+p|k)$ of the free component of a predicted output trajectory. As in all linear algorithms, we calculate this trajectory assuming control input increments equal to zero in the prediction horizon, $u(k|k) = u(k+1|k) = \cdots = u(k+5|k) = u_L(k-1)$:

$$\begin{aligned} y^0(k+1|k) &= g(x_L(k)) + d(k) \\ y^0(k+2|k) &= -a_1(k)[y^0(k+1|k) - y_L(k)] + b_2(k)[u(k-1) - u_L(k-2)] + \\ &\qquad + g(x_L(k)) + d(k) \\ &\vdots \\ y^0(k+6|k) &= -a_1(k)[y^0(k+5|k) - y_L(k)] + b_2(k)[u(k-1) - u_L(k-2)] + \\ &\qquad + g(x_L(k)) + d(k) \end{aligned}$$

where $d(k)$ is a disturbance estimate

$$d(k) = y(k) - y(k|k-1)$$

where $y(k|k-1)$ is the output value predicted for sampling instant $k$ on the basis of information from the previous instant $(k-1)$, using the linearized model.

To evaluate the dynamic matrix it is convenient to calculate coefficients of the step response of the linearized model, and then to use the formula (3.35). To calculate the step response we use the linearized equation of the dynamics (3.163) in the incremental form

$$\triangle y(k+1) = -a_1(k)\triangle y(k) + b_1(k)\triangle u(k-1) + b_2(k)\triangle u(k-2) \qquad (3.164)$$

where

$$\begin{aligned} \triangle y(k+1) &= y(k+1) - g(x_L(k)) \\ \triangle y(k) &= y(k) - y_L(k) \\ \triangle u(k-1) &= u(k-1) - u_L(k-1) \\ \triangle u(k-2) &= u(k-2) - u_L(k-2) \end{aligned}$$

It is most convenient to use the formulae (3.106). For the considered model we obtain 5 non-zero coefficients of the step response ($s_1 = 0$)

$$\begin{aligned} s_2 &= b_1(k) \\ s_3 &= -a_1(k)s_2 + b_1(k) + b_2(k) \\ s_4 &= -a_1(k)s_3 + b_1(k) + b_2(k) \\ s_5 &= -a_1(k)s_4 + b_1(k) + b_2(k) \\ s_6 &= -a_1(k)s_5 + b_1(k) + b_2(k) \end{aligned}$$

and the dynamic matrix $\mathbf{M}(k)$ (of dimensions $5 \times 4$) is calculated from (3.33). The forced component of the predicted outputs trajectory in the prediction horizon, $\Delta\mathcal{Y}(k)$, is given by $\Delta\mathcal{Y}(k) = \mathbf{M}(k)\Delta\mathcal{U}(k)$, where

$$\Delta\mathcal{U}(k) = [\Delta u(k|k)\ \Delta u(k+1|k)\ \Delta u(k+2|k)\ \Delta u(k+3|k)]^T$$

- *MPC-NPL algorithm*

The method of calculation of a nonlinear free output trajectory will first be presented. Assuming an initial trajectory of future control inputs in the form (3.156), elements of the nonlinear free output trajectory $\mathcal{Y}^0(k)$, calculated directly from the model (3.161), are given by

$$\begin{aligned} y^0(k+1|k) &= g(y(k), u(k-1), u(k-2)) + d(k) \\ y^0(k+2|k) &= g(y^0(k+1|k), u(k-1), u(k-1)) + d(k) \\ &\vdots \\ y^0(k+6|k) &= g(y^0(k+5|k), u(k-1), u(k-1)) + d(k) \end{aligned}$$

where $d(k)$ is a constant output disturbance estimate

$$d(k) = y(k) - g(y(k-1), u(k-2), u(k-3))$$

The dynamic matrix $\mathbf{M}(k)$, which corresponds to the incremental linearized model (3.164), is the same as in the MPC-NSL algorithm. □

### 3.5.5 MPC Algorithms Using Artificial Neural Networks

The use of artificial neural networks (ANNs) to modeling and control is now widely used and acknowledged. The ANNs are used first of all as universal nonlinear approximators to the construction of nonlinear process models, but also as nonlinear controllers [118]. It is not our aim in this book to present foundations and applications of ANNs; there is vast literature on the subject, see *e.g.*, [54]. Our goal is to describe the role of ANN modeling in construction of nonlinear predictive controllers, as originally presented in [136].

As discussed in the previous sections, the main problem in application of predictive controllers with nonlinear process models is the effectiveness and robustness of the nonlinear optimization procedure which, in turn, depends mainly on the properties of the process model used. Comprehensive process models, static and dynamic, are usually available as *simulation models*, *i.e.*, complex program packages delivering values of process outputs for submitted values of inputs. Optimization with such models is itself a complicated task, by no means suitable for on-line applications. In this situation, a proven way of proceeding is to approximate the simulation model by another one, with simpler structure and better numerical properties (complexity, continuity, differentiability, *etc.*). One of the best choices here is using simplified models built as ANNs. These models can then be used either as nonlinear process models in MPC-NO algorithms, or for linearization purposes in MPC-NSL or MPC-NPL algorithms. Both cases will be discussed in this section.

Let the single-input single-output process under consideration be described by the following nonlinear discrete-time equation of the form (3.85)

$$y(k) = g(u(k-\bar{\tau}), \ldots, u(k-n_B), y(k-1), \ldots, y(k-n_A)) \tag{3.165}$$

where $g$ is a continuously differentiable function and $\bar{\tau} \leq n_B$ represents time delay (including the discretization delay, *i.e.*, $\bar{\tau} = \tau + 1$). In the sequel it is assumed that the feedforward neural network with one hidden layer and linear output [54] is used as the function $g$ in (3.165). The structure of the neural network is depicted in Fig. 3.45. Output of the model can be expressed as

$$y(k) = w_2(0) + \sum_{i=1}^{K} w_2(i) v_i(k) = w_2(0) + \sum_{i=1}^{K} w_2(i) \varphi(z_i(k)) \tag{3.166}$$

where $z_i(k)$ is the sum of inputs and $v_i(k)$ is the output of the $i$-th hidden node, respectively, $\varphi$ is a scalar nonlinear transfer function, $K$ is the number

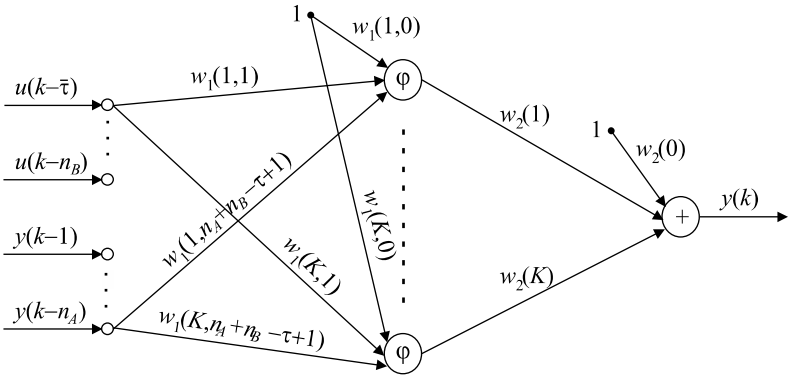


**Fig. 3.45.** Structure of the neural network

of hidden nodes. Recalling the input arguments of the general nonlinear model (3.165) one has

$$z_i(k) = w_1(i,0) + \sum_{j=1}^{I_u} w_1(i,j)u(k-\bar{\tau}+1-j) + \sum_{j=1}^{n_A} w_1(i,I_u+j)y(k-j) \quad (3.167)$$

The weights of the network are denoted by $w_1(i,j)$, $i = 1,\ldots,K$, $j = 0,\ldots,n_A+n_B-\bar{\tau}+1$, and $w_2(i)$, $i = 0,\ldots,K$, for the first and the second layer, respectively. The number of the network's input nodes depending on input signal $u$ is $I_u = n_B-\bar{\tau}+1$. Total number of weights is $(n_A+n_B-\bar{\tau}+2)K+K+1$.

**MPC Algorithms with Nonlinear Optimization and ANN Model**

In general, there are two methods of using neural models in MPC schemes with nonlinear optimization. In the first approach gradients of the cost function $J(k)$ are approximated numerically and the nonlinear optimization problem is solved on-line. In the second approach the structure of the neural network model is exploited [67, 72, 71, 118].

For simplicity of notation, the analysis will be shown for the cost function (3.2), *i.e.*, with one scalar weighting coefficient $\lambda$. Recall the vector of controller outputs/process control inputs over the control horizon $\mathcal{U}(k)$,

$$\mathcal{U}(k) = \begin{bmatrix} u(k|k) \\ \vdots \\ u(k+N_u-1|k) \end{bmatrix}$$

The cost function (3.2) in the form with control inputs $\mathcal{U}(k)$ instead of control input moves $\triangle\mathcal{U}(k)$ will now be more convenient (see Section 3.2.2 for definitions of other vectors)

$$J(k) = \|\mathcal{Y}^{sp}(k) - \mathcal{Y}(k)\|^2 + \lambda \left\|(\mathbf{I} + \mathbf{J}^{NO})\mathcal{U}(k) + \mathcal{U}^{NO}\right\|^2 \tag{3.168}$$

where the matrices $\mathbf{I}$ and $\mathbf{J}^{NO}$ are of dimension $N_u \times N_u$, the vector $\mathcal{U}^{NO}$ is of length $N_u$,

$$\mathbf{J}^{NO} = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ -1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & -1 & 0 \end{bmatrix}, \quad \mathcal{U}^{NO} = \begin{bmatrix} -u(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Differentiating (3.168) with respect to $\mathcal{U}(k)$ results in

$$\frac{\partial J(k)}{\partial \mathcal{U}(k)} = 2\left(\frac{\partial \mathcal{Y}(k)}{\partial \mathcal{U}(k)}\right)^T [\,\mathcal{Y}(k) - \mathcal{Y}^{sp}(k)\,] + 2\lambda \left(\mathbf{I} + \mathbf{J}^{NO}\right)^T [\,(\mathbf{I} + \mathbf{J}^{NO})\mathcal{U}(k) + \mathcal{U}^{NO}] \tag{3.169}$$

The matrix of dimension $(N - N_u + 1) \times N_u$, containing partial derivatives of the predicted outputs with respect to future control inputs is

$$\frac{\partial \mathcal{Y}(k)}{\partial \mathcal{U}(k)} = \begin{bmatrix} \dfrac{\partial y(k+N_1|k)}{\partial u(k|k)} & \cdots & \dfrac{\partial y(k+N_1|k)}{\partial u(k+N_u-1|k)} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial y(k+N|k)}{\partial u(k|k)} & \cdots & \dfrac{\partial y(k+N|k)}{\partial u(k+N_u-1|k)} \end{bmatrix} \tag{3.170}$$

The predictions $y(k+p|k)$ for $p = N_1, \ldots, N$ are calculated from the general prediction equation

$$y(k+p|k) = y(k+p) + d(k) \tag{3.171}$$

where the quantities $y(k+p)$ are calculated from the model. The above formulation assumes constant prediction of unmeasured disturbance $d(k)$ over the prediction horizon, as it was the case in the previous sections. Its value is estimated from the equation

$$\begin{aligned} d(k) &= y(k) - y(k|k-1) \\ &= y(k) - \left(w_2(0) + \sum_{i=1}^{K} w_2(i) v_i(k)\right) \end{aligned} \tag{3.172}$$

From the neural model (3.166) one has

$$y(k+p) = w_2(0) + \sum_{i=1}^{K} w_2(i)\varphi(z_i(k+p)) \tag{3.173}$$

Considering the prediction over the horizon $N$, the quantities $z_i(k+p)$, and consequently $y(k+p)$, will depend on some past process control inputs, on future control inputs, *i.e.*, decision variables of the controller optimization

problem, on measured values of the process outputs and on future output predictions. From (3.167) we have

$$\begin{aligned} z_i(k+p) = w_1(i,0) + \sum_{j=1}^{I_{uf}(p)} w_1(i,j)u(k-\bar{\tau}+1-j+p|k) + \\ + \sum_{j=I_{uf}(p)+1}^{I_u} w_1(i,j)u(k-\bar{\tau}+1-j+p) + \\ + \sum_{j=1}^{I_{yp}(p)} w_1(i,I_u+j)y(k-j+p|k) + \\ + \sum_{j=I_{yp}(p)+1}^{n_A} w_1(i,I_u+j)y(k-j+p) \end{aligned} \tag{3.174}$$

where $I_{uf}(p) = \max\{\min\{p-\bar{\tau}+1, I_u\}, 0\}$ is the number of the network's input nodes depending on future control input signals and $I_{yp}(p) = \min\{p-1, n_A\}$ is the number of the network's input nodes depending on output predictions. Because typically $N_u < N$, it can be noticed that

$$\sum_{j=1}^{I_{uf}(p)} u(k-\bar{\tau}+1-j+p|k) = \sum_{j=1}^{I_{N_u}(p)} u(k+N_u-1|k) + \sum_{j=I_{N_u}(p)+1}^{I_{uf}(p)} u(k-\bar{\tau}+1-j+p|k) \tag{3.175}$$

where $I_{N_u}(p) = \min\{\max\{p-\bar{\tau}-N_u+1, 0\}, I_u\}$. Taking into account (3.175), the equation (3.174) can be written as

$$\begin{aligned} z_i(k+p) = w_1(i,0) + \sum_{j=1}^{I_{N_u}(p)} w_1(i,j)u(k+N_u-1|k) + \\ + \sum_{j=I_{N_u}(p)+1}^{I_{uf}(p)} w_1(i,j)u(k-\bar{\tau}+1-j+p|k) + \\ + \sum_{j=I_{uf}(p)+1}^{I_u} w_1(i,j)u(k-\bar{\tau}+1-j+p) + \\ + \sum_{j=1}^{I_{yp}(p)} w_1(i,I_u+j)y(k-j+p|k) + \\ + \sum_{j=I_{yp}(p)+1}^{n_A} w_1(i,I_u+j)y(k-j+p) \end{aligned} \tag{3.176}$$

Taking into account (3.171) and (3.173), the entries of the matrix $\frac{\partial \mathcal{Y}(k)}{\partial \mathcal{U}(k)}$ given by (3.170), *i.e.*, partial derivatives of the predicted output signal with respect

to future control inputs, are calculated from

$$\frac{\partial y(k+p|k)}{\partial u(k+r|k)} = \sum_{i=1}^{K} w_2(i) \frac{d\varphi(z_i(k+p))}{dz_i(k+p)} \frac{\partial z_i(k+p)}{\partial u(k+r|k)} \tag{3.177}$$

Obviously

$$\frac{\partial z_i(k+p)}{\partial u(k+r|k)} = \frac{\partial y(k+p|k)}{\partial u(k+r|k)} = 0 \quad r \geq p - \bar{\tau} + 1 \tag{3.178}$$

It can be noted that decision variables of the algorithm affect only the first, the second and the fourth sum in (3.176). It can also be noted that only some of the output predictions are influenced by future inputs. Hence

$$\begin{aligned} \frac{\partial z_i(k+p)}{\partial u(k+r|k)} &= \sum_{j=1}^{I_{N_u}(p)} w_1(i,j) \frac{\partial u(k+N_u-1|k)}{\partial u(k+r|k)} + \\ &+ \sum_{j=I_{N_u}(p)+1}^{I_{uf}(p)} w_1(i,j) \frac{\partial u(k-\bar{\tau}+1-j+p|k)}{\partial u(k+r|k)} + \\ &+ \sum_{j=1}^{I_{ypf}(p)} w_1(i,I_u+j) \frac{\partial y(k-j+p|k)}{\partial u(k+r|k)} \end{aligned} \tag{3.179}$$

where $I_{ypf}(p) = \max\{\min\{p-\bar{\tau}, n_A\}, 0\}$ is the number of the network's input nodes depending on output predictions which are affected by future process control inputs. Obviously

$$\frac{\partial u(k+p|k)}{\partial u(k+r|k)} = \begin{cases} 0 & p \neq r \\ 1 & p = r \end{cases}$$

whereas the derivatives of predicted output signals with respect to future control inputs have to be calculated recursively.

The discussed method of calculating gradients of the predicted output trajectory with respect to the future process control inputs is used not only for obtaining the gradients of the cost function $J(k)$, but also for finding gradients of output constraints if they have to be taken into account. In some nonlinear optimization algorithms, for example SQP [5], the analytical Hessian matrix can be used. Unfortunately, this requires much more computational effort than calculating the gradients. That is why in the presented solution the optimization routine is provided with analytical gradients while the Hessian should be approximated, as it is done in most SQP practical implementations.

The extension of the presented MPC-NO algorithm with neural networks to MIMO systems is discussed in [67] and [71]. As the model of the process, $n_y = \dim y$ MISO (multi-input single-output) nonlinear models (neural networks) are used.

**Linearization-based MPC Algorithm with ANN Model**

As discussed earlier in this chapter, one of the most effective combinations is a nonlinear prediction of the free output trajectory together with the use of a linearized model for on-line optimization of future process input moves in the QP problem, *i.e.*, the MPC-NPL algorithm.

Defining a linearization point as the vector composed of past process input and output values

$$\overline{x}(k) = [\overline{u}(k-\bar{\tau}) \ \ldots \ \overline{u}(k-n_B-1) \ \overline{y}(k-1) \ \ldots \ \overline{y}(k-n_A)]^T \tag{3.180}$$

the linearized model has the form

$$\begin{aligned} y(k) = g(\overline{x}(k)) + \sum_{l=1}^{n_B+1} b_{l-1}(\overline{x}(k))(u(k-l) - \overline{u}(k-l)) + \\ - \sum_{l=1}^{n_A} a_l(\overline{x}(k))(y(k-l) - \overline{y}(k-l)) \end{aligned} \tag{3.181}$$

Taking into account the structure of the neural model and corresponding equations (3.166) and (3.167), the coefficients of the linearized model are calculated from

$$a_l(\overline{x}(k)) = -\frac{\partial g(\overline{x}(k))}{\partial y(k-l)} = -\sum_{i=1}^{K} w_2(i) \frac{d\varphi(z_i(\overline{x}(k)))}{dz_i(\overline{x}(k))} w_1(i, I_u + l), \quad l = 1, \ldots, n_A \tag{3.182}$$

$$b_{l-1}(\overline{x}(k)) = \begin{cases} 0, & l = 1, \ldots, \bar{\tau} - 1 \\ \dfrac{\partial g(\overline{x}(k))}{\partial u(k-l)} = \displaystyle\sum_{i=1}^{K} w_2(i) \frac{d\varphi(z_i(\overline{x}(k)))}{dz_i(\overline{x}(k))} w_1(i, l - \bar{\tau} + 1), & \\ & l = \bar{\tau}, \ldots, n_B + 1 \end{cases} \tag{3.183}$$

Let

$$\begin{aligned} a_l(k) &= a_l(\overline{x}(k)) \\ b_l(k) &= b_l(\overline{x}(k)) \end{aligned} \tag{3.184}$$

and redefine the variables

$$\begin{aligned} y(k) &= y(k) - g(\overline{x}(k)) \\ y(k-l) &= y(k-l) - \overline{y}(k-l), \quad l = 1, \ldots, n_A \\ u(k-l) &= u(k-l) - \overline{u}(k-l), \quad l = 1, \ldots, n_B + 1 \end{aligned} \tag{3.185}$$

then the linear approximation of the model (3.165), obtained at sampling instant $k$, can be expressed as

$$\mathbf{A}(k, z^{-1}) y(k) = \mathbf{B}(k, z^{-1}) u(k) \tag{3.186}$$

where, consequently, as the total time delay $\bar{\tau} \geq 1$ is used

$$\mathbf{A}(k,z^{-1}) = 1 + a_1(k)z^{-1} + \ldots + a_{n_A}(k)z^{-n_A} \qquad (3.187)$$
$$\mathbf{B}(k,z^{-1}) = b_0(k)z^{-1} + \ldots + b_{n_B}(k)z^{-n_B-1}$$

It can be noted that the linearization point given by (3.180), and hence the coefficients $a_l(k)$ and $b_l(k)$, are not influenced by the most recent output value $y(k)$, which is available as current measurement. It may be crucial in the case of fast processes. Therefore, it is then recommended to use the linearization point

$$\overline{x}(k) = [\overline{u}(k-\bar{\tau}+1) \ \ldots \ \overline{u}(k-n_B) \ \overline{y}(k) \ \ldots \ \overline{y}(k-n_A+1)]^T \qquad (3.188)$$

If $\bar{\tau} = 1$, for linearization purposes $u(k) = u(k-1)$ or $u(k) = u(k|k-1)$ can be chosen. The MPC-NPL algorithm using the linearization point (3.180) will be denoted MPC-NPL1, whereas the one which uses (3.188) MPC-NPL2.

The nonlinear free output trajectory $y^0(k+p|k)$, $p = 1, \ldots, N$, is calculated recursively from the general prediction equation (3.171), taking into account the output of the neural model given by (3.173) and the constant output disturbance prediction (3.172)

$$y^0(k+p|k) = w_2(0) + \sum_{i=1}^{K} w_2(i)\varphi(z_i^0(k+p)) + d(k) \qquad (3.189)$$

The quantities $z_i^0(k+p)$ are determined from (3.174) assuming no changes in control input signal from sampling instant $k$ to the end of the prediction horizon, and replacing predicted output signals by corresponding values of the free output trajectory,

$$u(k+p|k) = u(k-1) \quad p \geq 0$$
$$y(k+p|k) = y^0(k+p|k) \quad p \geq 1$$

hence

$$\begin{aligned} z_i^0(k+p) = w_1(i,0) &+ \sum_{j=1}^{I_{uf}(p)} w_1(i,j)u(k-1) \ + \\ &+ \sum_{j=I_{uf}(p)+1}^{I_u} w_1(i,j)u(k-\bar{\tau}+1-j+p) \ + \\ &+ \sum_{j=1}^{I_{yp}(p)} w_1(i,I_u+j)y^0(k-j+p|k) \ + \\ &+ \sum_{j=I_{yp}(p)+1}^{n_A} w_1(i,I_u+j)y(k-j+p) \end{aligned} \qquad (3.190)$$

The extension of the presented MPC-NPL algorithm with neural networks to MIMO systems is discussed in [67].

The stability of the presented MPC-NPL and MPC-NO algorithms with neural models can be achieved by properly tuning the weighting coefficient $\lambda$ in the cost function $J(k)$. Both these algorithms can be combined with the stabilising dual-mode approach [74], [67], originally developed by H. Michalska and D.Q. Mayne [97]. Stability of MPC algorithms will be discussed in Section 3.6.1.

To reduce the computational complexity of the nonlinear MPC algorithm, several original neural-network based solutions have been suggested. In general, these approaches can be divided into two groups:

- Special structures of ANN models designed to make the on-line MPC controller optimization problem computationally simpler [152, 80, 110].
- Approximate explicit (without on-line optimization) MPC algorithms combined with neural networks [109, 108, 107, 56, 53, 149].

Presentation of these, often specific approaches is beyond the scope of this book, the interested reader is referred to [136] for a review.

### 3.5.6 Comparative Simulation Studies

In this section three simulation studies will be presented, illustrating applications of different nonlinear MPC algorithms, including also cases with process modeling by neural networks. Comparisons with linear MPC algorithms will also be given.

**Example 3.8**
Predictive control of a jacketed continuously-stirred tank reactor with polymerization reaction described in [34] will be considered. The reaction under consideration is the free-radical polymerization of methyl methacrylate with azo-bis-isobutyronitrile as initiator and toluene as solvent. Under certain simplifying assumptions, see [34] (ideal mixing, constant volume, constant temperature equal to 335 K, constant heat capacity, *etc.*), the following reactor equations were obtained:

$$\begin{aligned}
\dot{x}_1 &= 10(6 - x_1) - 2.4568 x_1 \sqrt{x_2} \\
\dot{x}_2 &= 80u - 10.1022 x_2 \\
\dot{x}_3 &= 0.0024121 x_1 \sqrt{x_2} + 0.112191 x_2 - 10 x_3 \\
\dot{x}_4 &= 245.978 x_1 \sqrt{x_2} - 10 x_4 \\
y &= x_4 / x_3
\end{aligned}$$

where $x_1 = C_m$ is the monomer concentration, $x_2 = C_1$ is the initiator concentration, and the controlled output $y$ is the NAMV (number-average molecular weight $[kg/kmol]$), which is a quotient of the fourth and third state variable, $y = x_4/x_3$. The initiator flow rate $F_I$ is the process manipulated input $u$. A diagram of the reactor control system is presented in Fig. 3.46.

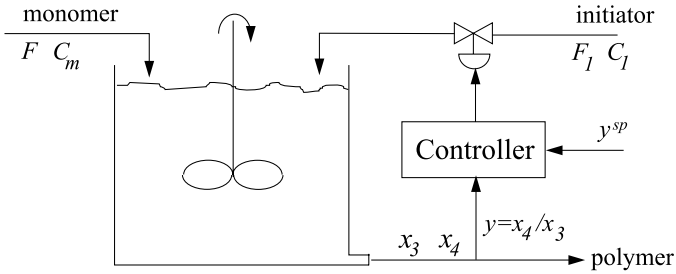


**Fig. 3.46.** The polymerization reactor control structure

The presented process is strongly nonlinear, with nonlinear statics and dynamics. Let us assume an operating point as in [34]: $x_{10} = 5.50677$, $x_{20} = 0.132906$, $x_{30} = 0.0019752$, $x_{40} = 49.3818$, $u_0 = 0.016783$, thus $y_0 = 25000$. Constraints on process input amplitude $u_{\min} = 0,0035$, $u_{\max} = 0.0336$ should also be considered.

Simulations of the control system with the following three predictive controllers were investigated:

- A linear GPC controller in a numerical version (with on-line optimization) designed for the given operating point.
- A MPC-NPL controller, with a nonlinear prediction of the free output trajectory calculated using a discretized version of the original nonlinear model and quadratic optimization with a linearized model, at each sampling instant.
- A full nonlinear MPC-NO controller, with both prediction and optimization using a discretized nonlinear model.

The following parameters were assumed for the controllers:

- prediction horizon $N = 10$,
- control horizon $N_u = 2$,
- weighting coefficients $\mathbf{\Psi}(p) = \mathbf{I}$, $\mathbf{\Lambda}(p) = \lambda\mathbf{I}$, with $\lambda = 700$,
- sampling period $T_p = 1.8$ $[min]$.

Selected results of simulation experiments [67], performed using programs of the REGZA package [68] are shown in Figures 3.47 - 3.50.

Figures 3.47 and 3.48 present trajectories obtained for the step-change in the set-point to the value 28000. Differences in the operation of the controllers are already visible, especially in the trajectories of the process control input signal. The change of the operating point is in the direction of a larger process gain, thus the GPC controller, designed for the operating point with the smaller gain value, shows the fastest operation, but still operates in a stable and acceptable way.

Figures 3.49 and 3.50 show trajectories after a larger step-change in the set-point value, to 32000. Differences in the operation of the controllers are

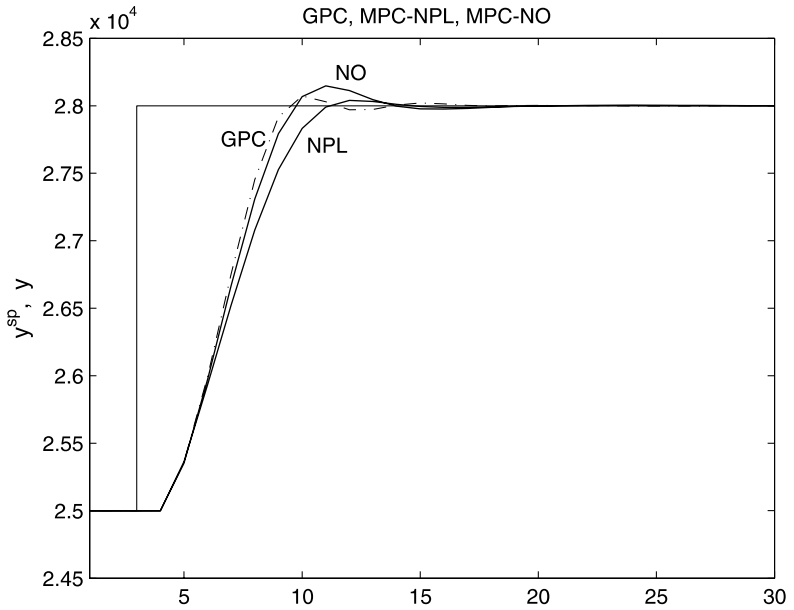


**Fig. 3.47.** Trajectories of the output variable in the polymerization reactor control systems for a step-change in the set-point to 28000

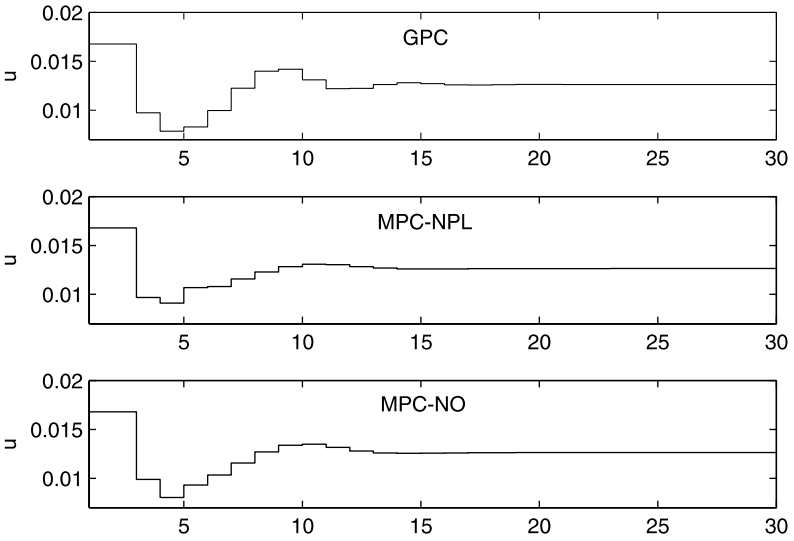


**Fig. 3.48.** Trajectories of the control input signal in the polymerization reactor control systems for a step-change in the set-point to 28000

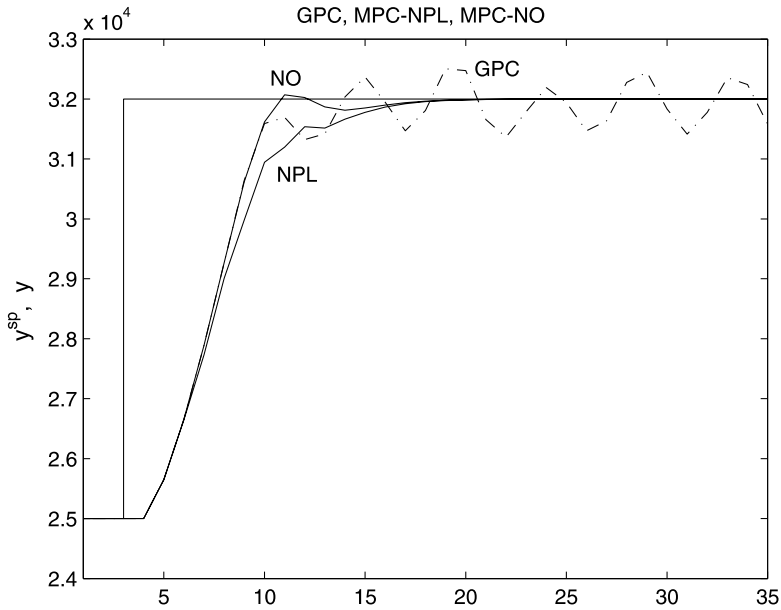


**Fig. 3.49.** Trajectories of the output variable in the polymerization reactor control systems for a step-change in the set-point to 32000

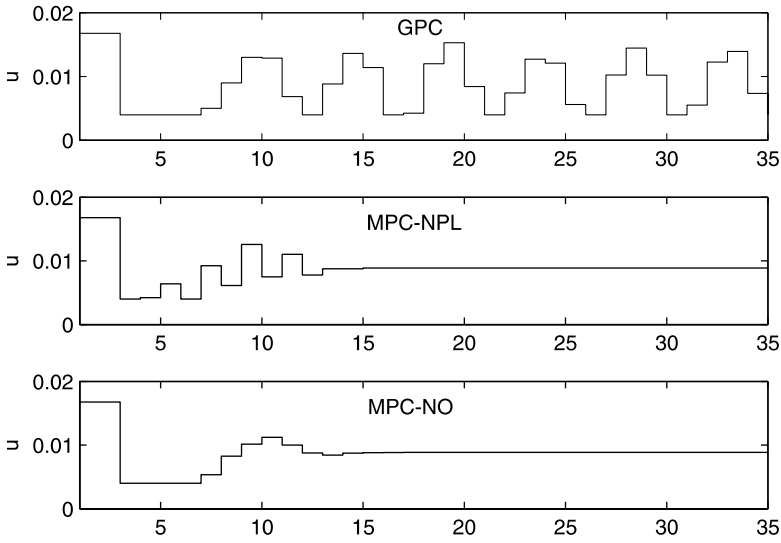


**Fig. 3.50.** Trajectories of the control input signal in the polymerization reactor control systems for a step-change in the set-point to 32000

now very distinct, both in trajectories of the controlled variable and of the control input signal. The system with the GPC controller became unstable, while the system with the simplest version of the MPC-NPL control algorithm operates slower than with the MPC-NO one, yet in a stable and quite efficient way. □

**Example 3.9**
Selected results of simulation experiments with two-dimensional predictive control of a nonlinear distillation column presented earlier in Example 2.3 in Section 2.1 will be given.

Concentrations $x_d$ and $x_b$ of the product in the distillate $D$ and in the bottom flow $B$, respectively, are the controlled variables (controlled outputs) $y$. The process control inputs $u$ are the reflux flow rate $R$ and the heating steam flow rate $V$. Assuming the sampling period $T_p = 10\ [min]$ two predictive controllers were designed [67]:

- A linear GPC controller with the structure of the models (3.114),

$$y_1(k) = -a_1^1 y_1(k-1) + b_0^{1,1} u_1(k-1) + b_0^{1,2} u_2(k-1)$$
$$y_2(k) = -a_1^2 y_2(k-1) + b_0^{2,1} u_1(k-1) + b_0^{2,2} u_2(k-1)$$

  designed for an operating point defined by the values

  $R_0 = 33.34\ [kmol/h]$
  $V_0 = 83.34\ [kmol/h]$
  where
  $y = [x_d\ x_b]^T$, $u = [R\ V]^T$.

- A MPC-NO controller, with a nonlinear prediction and optimization using the full nonlinear model of the column.

It was assumed that $\mathbf{\Psi}(p) = \mathbf{I}$, $\mathbf{\Lambda}(p) = \lambda \mathbf{I}$. Simulation tests were performed for different prediction horizons and different values of $\lambda$. Representative trajectories obtained for both algorithms for $N = 6$, $N_u = 2$ and $\lambda = 0.05$ are presented in Figures from 3.51 to 3.54.

It follows from the comparison that in spite of the nonlinearity of the column, which is obvious when looking at the static characteristics presented in Example 2.3, the GPC controller operates quite well. The MPC-NO controller (with full nonlinear optimization) is slightly quicker and suppresses interactions between the outputs in a better way.

It was assumed in the presented simulations, unlike in other simulation examples, that the predictive controllers know in advance about future changes of the set-point values which are to happen in the prediction horizon. An anticipative character of the control is then clearly visible, the controllers react ahead to the changes which are about to occur.

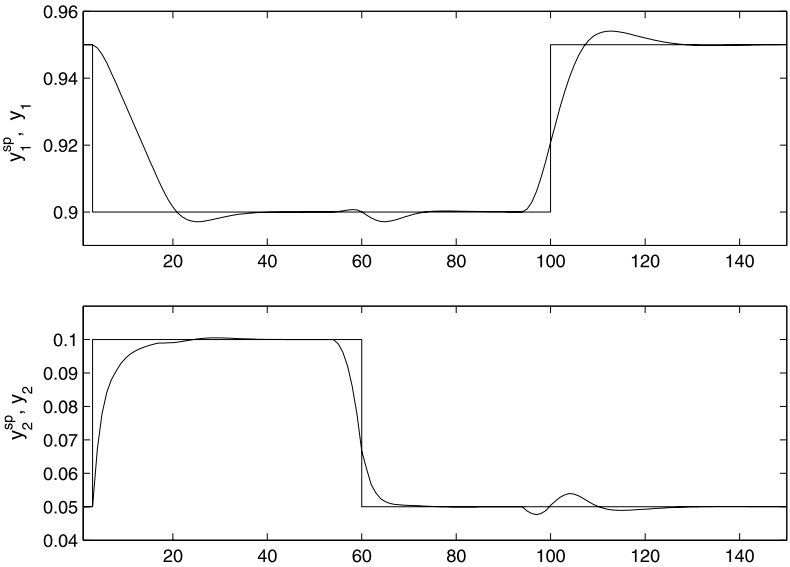


**Fig. 3.51.** Output trajectories in the GPC control system of the column

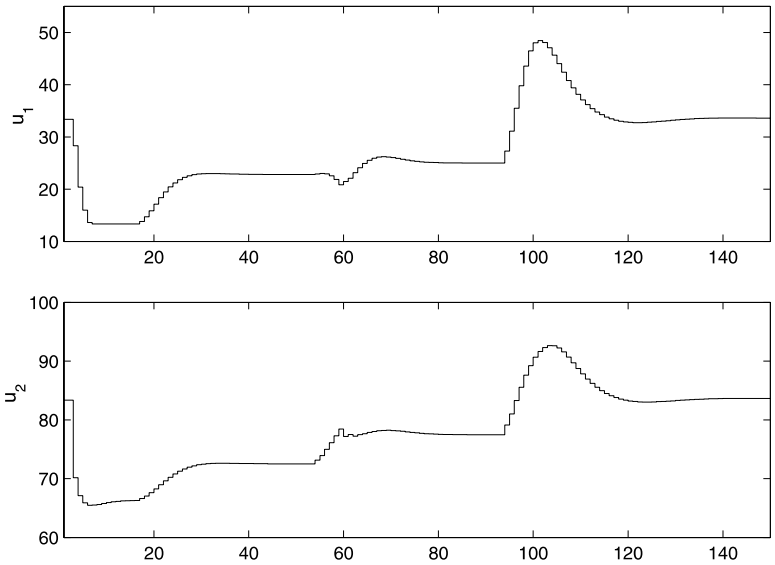


**Fig. 3.52.** Input trajectories in the GPC control system of the column

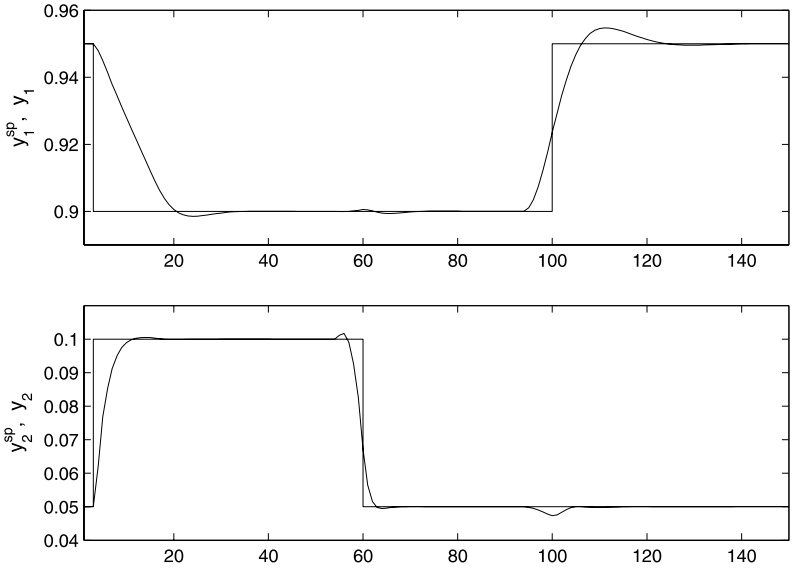


**Fig. 3.53.** Output trajectories in the MPC-NO control system of the column

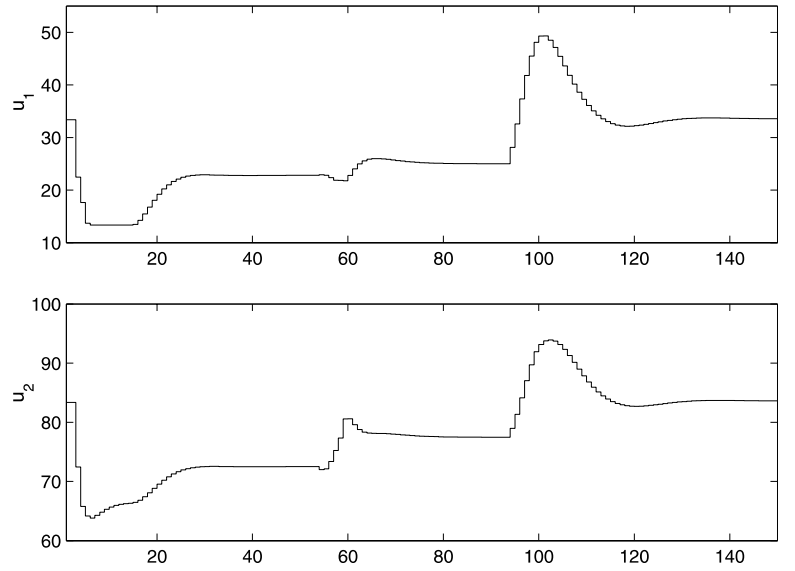


**Fig. 3.54.** Input trajectories in the MPC-NO control system of the column □

**Example 3.10**
The plant under consideration is a high purity, high pressure $(1,93\,MPa)$ ethylene-ethane distillation column shown in Fig. 3.55, [67, 136]. The feed stream consists of ethylene (approx. 80%) ethane (approx. 20%) and traces of hydrogen, methane and propylene. The product of the distillation is ethylene which can contain up to 1000 $[ppm]$ (*parts per million*) of ethane. The main problem is to develop a constraint controller which would be able to increase the impurity level relatively fast when the composition changes in the feed stream are relatively small. Reducing the purity of the product, of course taking into account the technological limit, results in decreasing energy consumption. The production scale is very big; the nominal value of the product stream flow rate is 43 $[tons/h]$. The column has 121 trays, the feed stream is delivered to tray number 37.

Two fast single-loop PID controllers (denoted as LC) are used to maintain the levels in reflux tank and bottom product tank. Yet another PID controller (denoted as TC) is also used to control the temperature on the tray number 13. The PID controllers comprise the basic control layer. As far as the supervisory constraint MPC controller is concerned, the control loop has one process input variable $r$, which is the ratio $r = \frac{R}{P}$, where $R$ is the reflux stream flow rate delivered to the column by the top tray and $P$ is the product stream flow rate taken from the tray number 110. There is one output variable $z$, which represents the impurity level in the product (ethylene). Sampling interval of

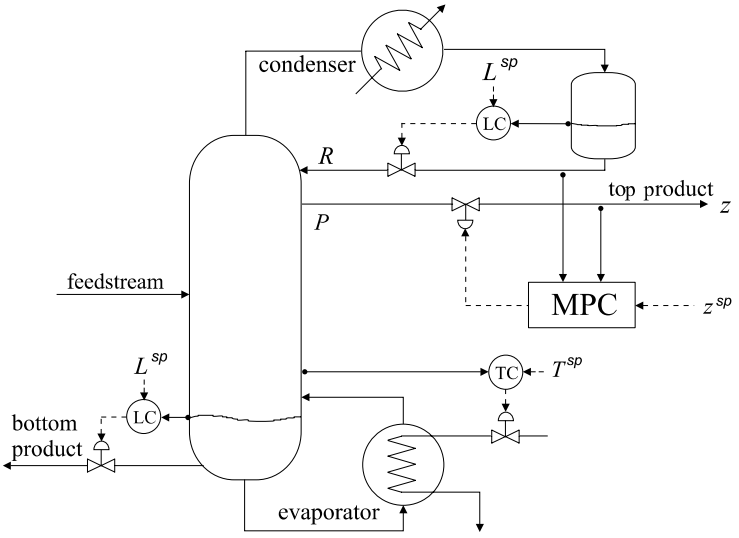


**Fig. 3.55.** High-purity high-pressure ethylene-ethane distillation column control system structure, Example 3.10

the MPC algorithm is relatively long (slow composition analyzer), equal to $Tp = 40$ $[min]$.

Four models of the plant were used. The first and most accurate one was used as the process representation in the closed-loop during the simulations and was based on technological considerations [67]. An identification procedure was carried out, as a result two linear models for different operating points and a nonlinear neural network model were obtained. For the empirical models $n_A = 1$, $\tau = n_B = 3$. The horizons were set to $N = 10$, $N_u = 3$, the weighting coefficient $\lambda = 2$. In all the simulations it was assumed that at sampling instant $k = 1$ the set-point value is changed from 100 $[ppm]$ to 350, 600 and 850 $[ppm]$, respectively. Due to technological reasons the following constraints were imposed on the reflux ratio: $r^{min} = 4.051$, $r^{max} = 4.4571$.

First, the algorithms based on two linear models were developed. The first linear model is valid for a "low" impurity level and the resulting control algorithm works well in this region, but exhibits unacceptable oscillatory behavior for medium and large set-point changes, as it is shown in Fig. 3.56. On the contrary, the second linear model captures the process properties for a "high" impurity level and the closed-loop response is fast enough for the biggest set-point change but is very slow for smaller ones, as it is shown in Fig. 3.57.

Simulation results with the MPC-NPL algorithms with a neural network model are depicted in Fig. 3.58. Both algorithms work well for all three set-point changes, the NPL1 algorithm is slightly slower than NPL2 (recall these algorithms differ slightly in a way the linearization points are chosen, see the preceding section). Simulation results with the MPC-NO algorithm with the neural network model are shown in Fig. 3.59. In comparison with suboptimal linearization-based algorithms, the nonlinear optimization leads to slightly faster closed-loop responses.

In practice, big changes in the input variable $r$ are not allowed because of technological and safety reasons (high pressure, big production scale). That

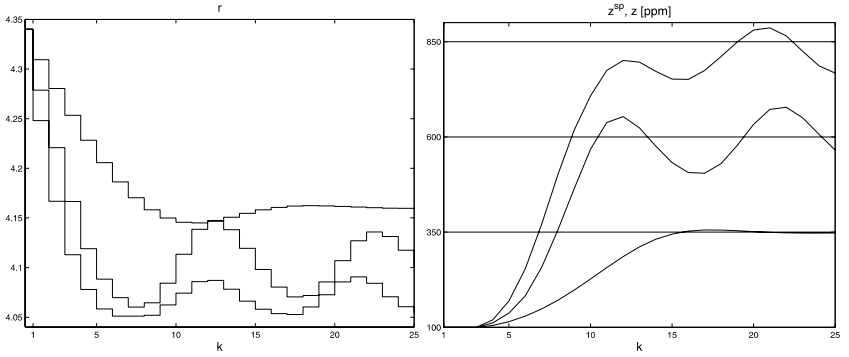


**Fig. 3.56.** Simulation results of the ethylene distillation column with MPC algorithm based on a linear model for "low" impurity level

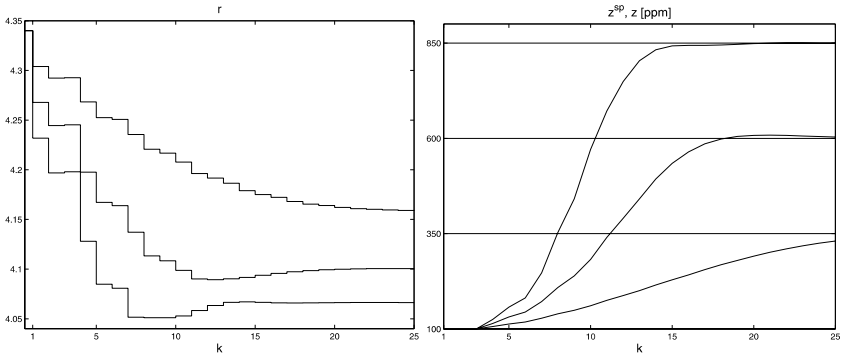


**Fig. 3.57.** Simulation results of the ethylene distillation column with MPC algorithm based on a linear model for "high" impurity level

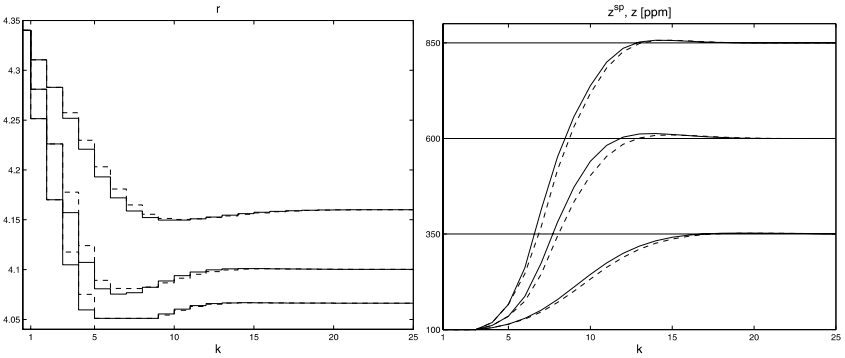


**Fig. 3.58.** Simulation results of the ethylene distillation column with MPC-NPL1 (dashed line) and MPC-NPL2 (solid line) algorithms with neural network model

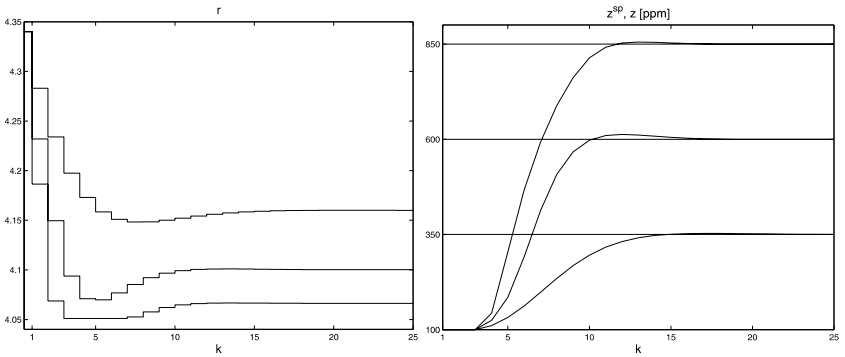


**Fig. 3.59.** Simulation results of the ethylene distillation column with MPC-NO algorithm with neural network model

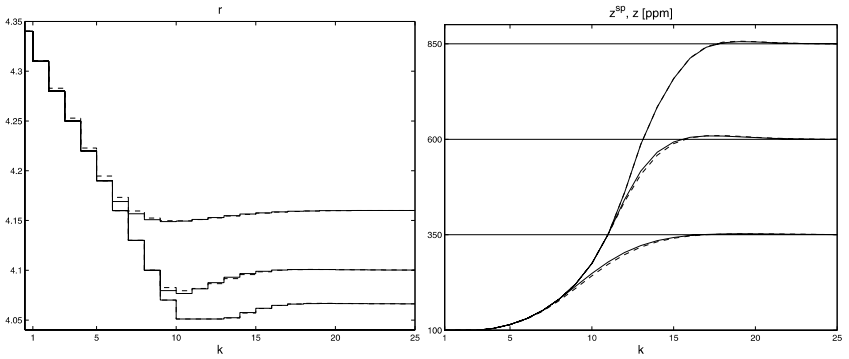


**Fig. 3.60.** Simulation results of the ethylene distillation column with MPC-NPL2 (dashed) and MPC-NO (solid) algorithms, neural network model and $\Delta r_{max} = .03$

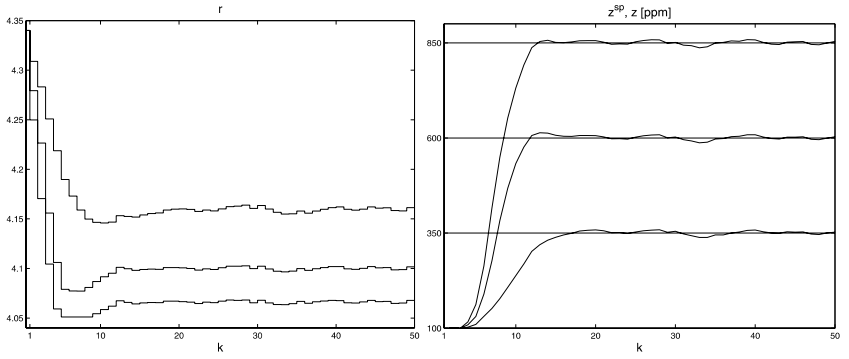


**Fig. 3.61.** Simulation results of the ethylene distillation column with MPC-NPL2 algorithm with neural network model and unmeasured disturbances

is why an additional constraint $\Delta r^{max} = 0.03$ was used. Fig. 3.60 compares simulation results of the MPC-NPL2 and MPC-NO algorithms, with the neural network model. Although the constraint significantly slows down the closed-loop responses, the MPC-NO algorithm is still somewhat faster.

Simulation results of the MPC-NPL2 algorithm and with unmeasured stochastic disturbances added to the process are presented in Fig. 3.61. Such disturbances are unavoidable in industry. □

### 3.5.7 Fuzzy MPC (FMPC) Numerical Algorithms

A Takagi-Sugeno (TS) fuzzy model is an example of a nonlinear model. If we use this model in the structure of the MPC-NO algorithm (which can then be called the FMPC-NO algorithm), then at each sampling instant the optimization problem (3.155) is solved, the problem with a nonlinear objective function, linear constraints on process inputs and nonlinear constraints on

controlled outputs, if such exist. The predicted trajectory of controlled outputs is then calculated using the nonlinear TS fuzzy model.

The key problem here is a right selection of a numerical optimization procedure, so as to ensure its reliable and sufficiently fast operation. Formulation of the recommendations is not easy here, only general indications concerning a choice of minimization procedures corresponding to specific features of nonlinear problems can be given, known from theory and practice of optimization, see *e.g.*, [43, 9, 129]. Numerical SQP-type (Sequential Quadratic Programming) procedures are the ones which are most frequently recommended. In a simpler case of one-dimensional control the problem of selection of the optimization procedure becomes less critical, see *e.g.*, [42]. A discussion concerning applications of optimization methods in nonlinear predictive control algorithms can be found *e.g.*, in [1, 82]. For stability analysis of the FMPC-NO algorithms, it is possible to use general approaches proposed in the literature for nonlinear models, which will be discussed further on, in Section 3.6.1.

The situation is much more advantageous in cases of nonlinear FMPC *suboptimal algorithms*, at each sampling instant performing linearization and using a linearized form of the fuzzy model for prediction and on-line optimization (FMPC-NSL algorithms), or performing prediction on a nonlinear model and optimization on its current linearized version (FMPC-NPL algorithms). This explains a large practical significance of these solutions, more so because differences in the operation of control systems resulting from suboptimality of the controllers can often be not very significant, especially in cases where a nonlinear prediction of the free output trajectory is used (the NPL structure).

It will be presented now that for the FMPC suboptimal algorithms using linearized models, it is important to have the TS fuzzy process model with *identical structure of all the consequents of the fuzzy rules*. Denoting the vector of variables of a general process model by $x$, $x = [x_1, x_2, \ldots, x_n]^T$, where elements of $x$ can consist of state variables or process inputs and outputs (current and delayed), the rules $\mathrm{R}^i$ of a TS fuzzy model can be written is the following form (see Chapter 2)

$$\begin{aligned} \mathrm{R}^i: \ \text{IF} \ \ & x_1(k) \text{ is } A_1^i \ \text{ and } \ \cdots \ \text{ and } \ x_n(k) \text{ is } A_n^i \\ & \text{THEN } \ y^i(k+1) = a_0^i + a_1^i x_1(k) + \cdots + a_n^i x_n(k) \end{aligned}$$

The output of the fuzzy model is given by the formula

$$y(k+1) = \frac{\sum_{i=1}^{r} w^i(k) y^i(k+1)}{\sum_{l=1}^{r} w^l(k)} = \sum_{i=1}^{r} \tilde{w}^i(k)[a_0^i + \sum_{j=1}^{n} a_j^i x_j(k)] \qquad (3.191)$$

where $r$ denotes the number of rules, $w^i(k)$ - levels of activation of the fuzzy model at sampling instant $k$ (at a point $x(k)$), while $\tilde{w}^i(k)$ denote normalized levels of activation, see Section 2.1.2. Due to affinity and identical structure of all consequents of the rules we can present (3.191) as follows

$$y(k+1) = a_0(k) + \sum_{j=1}^{n} a_j(k) x_j(k) \tag{3.192}$$

where

$$a_j(k) = \sum_{i=1}^{r} \tilde{w}^i(k) a_j^i, \quad j = 0, ..., n \tag{3.193}$$

In this way we obtain a fuzzy model in the form of an *affine model with variable coefficients*, with values depending on a current process state. If at the current sampling instant the process state is $x(k)$, then the simplest and most natural way of generating a local linear model, to be used in the optimization problem of a predictive control algorithm, is to take the model with the coefficients as in (3.192), with actual values of these coefficients corresponding to levels of activation of fuzzy rules, calculated for the current process state.

**FMPC-NSL (Fuzzy MPC-NSL) Algorithms**

For a TS fuzzy model with all rule consequents of identical structure, realization of a FMPC-NSL algorithms is natural and relatively easy. At each step of the algorithm a linear model stemming from (3.192) is calculated. Depending on the assumed method of modeling it will be, *e.g.*, a model defined by coefficients of step responses, a model in the form of discrete difference equations or linear state equations. As a result, at each sampling instant the control input signal will be generated as in the linear MPC algorithm chosen, DMC, GPC or MPCS, respectively – leading to nonlinear algorithms which can be called FDMC-NSL, FGPC-NSL or FMPSC-NSL, respectively. Formulae defining elements of free and forced output trajectories will be as in the DMC, GPC or MPCS algorithms, but with variable coefficients changing from one sampling instant to the next, corresponding to the changing coefficients of a linear model used.

Consider first the FDMC-NSL algorithm. According to the assumed structure of fuzzy modeling, the model domain will be divided into $r$ fuzzy subdomains (multivariable fuzzy sets, in general). One local linear process model will correspond to each of these sub-domains, given by coefficients of the step response $s_j^i$, $j = 1, ..., D$, $i = 1, ..., r$, where $D$ is a common, for all linear models, number of elements of the step responses (see Section 3.2.1). The only elements of the DMC algorithm connected with the model are matrices $\mathbf{M}^P$ and $\mathbf{M}$, which are used to define the free and forced output trajectories, see (3.38) and (3.39) in Section 3.2.2. In the case of the considered fuzzy algorithm, they will be built from elements of the step response, with values dependent on the current sampling instant $k$, $s_j = s_j(k) = \sum_{i=1}^{r} \tilde{w}^i(k) s_j^i$, where $\tilde{w}^i(k)$ denote normalized activation levels of fuzzy rules. At each subsequent sampling instant $k$ the control input increments will be generated on the basis of the DMC algorithm with matrices $\mathbf{M}^P = \mathbf{M}^P(k)$ and $\mathbf{M} = \mathbf{M}(k)$, built from elements $s_j(k)$ (in a MIMO case: $\mathbf{S}_j(k)$).

In the *FGPC-NSL algorithm*, values $a_j = a_j(k)$ and $b_j = b_j(k)$ of coefficients of difference equations modeling the process will depend on activation

levels of fuzzy rules of a fuzzy model at sampling instant $k$. On the basis of these equations matrices $\mathbf{F} = \mathbf{F}(k)$ and $\mathbf{G}^{PG} = \mathbf{G}^{PG}(k)$ will be constructed, which are used for calculation of a free output trajectory in the GPC algorithm, and the dynamic matrix $\mathbf{M} = \mathbf{M}(k)$ $(= \mathbf{G}^{FG}(k))$ needed to calculate a forced output trajectory, see Section 3.3. Elements of the free output trajectory can also be calculated recurrently using (3.110).

Similarly, in the *FMPCS-NSL algorithm* coefficients of matrices of linear state and output equations will depend on activation levels of fuzzy rules at each sampling instant, $\mathbf{A} = \mathbf{A}(k)$, $\mathbf{B} = \mathbf{B}(k)$, $\mathbf{C} = \mathbf{C}(k)$. These matrices are then further used to construct matrices $\widetilde{\mathbf{A}} = \widetilde{\mathbf{A}}(k)$, $\mathbf{V} = \mathbf{V}(k)$ and $\mathbf{M}_x = \mathbf{M}_x(k)$ needed for state and output predictions at each sampling instant $k$, see Section 3.4. However, the full state vector is usually unavailable, therefore these predictions then depend on estimated state and disturbance values. In a nonlinear case, a nonlinear state observer should be used. The design of such observers is not as simple as in the linear case. However, if we use a nonlinear TS fuzzy process model based on local models in the form of linear state equations, then a nonlinear fuzzy state observer constructed in an analogous way, as a TS fuzzy system based on local linear observers, can be recommended [132]. An efficient way of an observer design can also be the application an an extended Kalman filter [77].

### FMPC-NPL (Fuzzy MPC-NPL) Algorithms

Let us now move on to FMPC-NPL algorithms, with nonlinear prediction of the free output trajectory and linearized models used for optimization of the forced output trajectory. These algorithms are slightly more difficult in implementation than the FMPC-NSL ones. Let us begin with the case of linear models in the form used in the GPC algorithm.

*FGPC-NPL Algorithm*

A SISO process case will now be considered, as in Section 3.3.1. Assume that a fuzzy process model consists of $r$ rules, and that all rule consequents are linear models in the form of discrete difference equations corresponding to the structure (3.105). A fuzzy model for a nonlinear output prediction can then be written in the form of a set of rules

$$\begin{aligned} \mathrm{R}^i : \mathrm{IF} \quad & y(k) \text{ is } A_0^i \text{ and } y(k-1) \text{ is } A_1^i \text{ and } \cdots \text{ and } y(k-n_R) \text{ is } A_{n_R}^i \\ & \text{and } u(k) \text{ is } B_0^i \text{ and } u(k-1) \text{ is } B_1^i \text{ and } \cdots \text{ and } u(k-m_R) \text{ is } B_{m_R}^i \\ \text{THEN} \quad & y^i(k+1) = -a_1^i y(k) - \cdots - a_{n_A}^i y(k-n_A+1) + \\ & \quad + b_0^i u(k) + b_1^i u(k-1) + \cdots + b_{n_B}^i u(k-n_B) \end{aligned} \tag{3.194}$$

where $i$ indexes rules and at the same time sub-domains (fuzzy sets, multivariable in general) of the TS model, $A_j^i \in \mathbb{Y}_j$, $B_j^i \in \mathbb{U}_j$, while $a_j^i$ and $b_j^i$ are coefficients of functions in rule consequents, $i = 1, ..., r$, and $u(k) = u(k|k)$.

Elements of each of the sets $\mathbb{Y}_j = \{Y_{j1}, ..., Y_{jr_{yj}}\}$ are fuzzy sets covering the area of the variable $y(k-j)$, $j = 0, \ldots, n_R$, analogously $\mathbb{U}_j = \{U_{j1}, ..., U_{jr_{uj}}\}$ for $u(k-j)$, $j = 0, \ldots, m_R$. The presented description is most general, we frequently have $\mathbb{Y}_0 = \cdots = \mathbb{Y}_{n_R} = \mathbb{Y}$, $\mathbb{Y} = \{Y_1, ..., Y_{r_y}\}$, *i.e.*, partitions of the domain of $y(k)$ are the same, independent on time delays, and analogously $\mathbb{U}_1 = \cdots = \mathbb{U}_{m_R} = \mathbb{U}$, $\mathbb{U} = \{U_1, ..., U_{r_u}\}$ (see Sections 2.1.2 and 2.2.2).

The set of rules is complemented by the standard formula for a final conclusion, namely for the model output

$$y(k+1) = \sum_{i=1}^{r} \tilde{w}^i(k) y^i(k+1) \tag{3.195}$$

where $\tilde{w}^i(k)$ are normalized activation levels of the rules (3.194). Note that activation levels of the rules at sampling instant $k$ depend on values of $n_R+1$ outputs ($y(k)$, $y(k-1)$, ..., $y(k-n_R)$ and $m_R+1$ inputs $u(k)$, $u(k-1)$, ..., $u(k-m_R)$. Precisely, they depend on grades of membership of these outputs and inputs to the fuzzy sets $A^i_j$ and $B^i_j$, where, in general, $n_R \neq n_A$ and $m_R \neq n_B$.

A concise description of the model (3.194), (3.195), compliant with the general formula (3.192), is

$$\begin{aligned} y(k+1) = &-a_1(k)y(k) - \cdots - a_{n_A}(k)y(k-n_A+1) + \\ &+b_0(k)u(k) + b_1(k)u(k-1) + \cdots + b_{n_B}(k)u(k-n_B) \end{aligned} \tag{3.196}$$

where

$$a_j(k) = \sum_{i=1}^{r} \tilde{w}^i(k) a^i_j, \quad b_j(k) = \sum_{i=1}^{r} \tilde{w}^i(k) b^i_j \tag{3.197}$$

Assuming a constant output disturbance model, with a disturbance estimate $d(k)$

$$d(k) = y(k) - \left[ -\sum_{j=1}^{n_A} a_j(k-1)y(k-j) + \sum_{j=0}^{n_B} b_j(k-1)u(k-1-j) \right]$$

we can start with formulating rules of the *nonlinear prediction*.

The output $y^0(k+1|k)$ predicted at sampling instant $k$ for the instant $k+1$ can be evaluated directly from (3.194) and (3.195), or equivalently from (3.196). Thus we obtain

$$\begin{aligned} y^0(k+1|k) = &-a_1(k)y(k) - \cdots - a_{n_A}(k)y(k-n_A+1) + \\ &+b_0(k)u^0(k|k) + b_1(k)u(k-1) + \cdots + b_{n_B}(k)u(k-n_B) + d(k) \end{aligned} \tag{3.198}$$

where $u^0(k|k) = u(k-1)$ is the first element of an initial trajectory of control inputs over the control horizon $\mathcal{U}^0(k)$, see (3.156).

Output values predicted for the next sampling instants will be calculated using, recurrently, the one-step ahead prediction based on (3.194) and (3.195). Therefore, the output predicted for sampling instant $k+2$ will be evaluated from the formulae

$$\begin{aligned}\text{IF}\ \ & y^0(k+1|k) \text{ is } A_0^i \ \text{ and } \ y(k) \text{ is } A_1^i \ \text{ and } \cdots \text{ and } \ y(k-n_R+1) \text{ is } A_{n_R}^i \\ & \text{and } \ u^0(k+1|k) \text{ is } B_0^i \ \text{ and } \ u^0(k|k) \text{ is } B_1^i \ \text{ and } \cdots \\ & \qquad \cdots \text{ and } \ u(k-m_R+1) \text{ is } B_{m_R}^i \\ \text{THEN}\ \ & y^{0i}(k+2|k) = -a_1^i y^0(k+1|k) - a_2^i y(k) - \cdots - a_{n_A}^i y(k-n_A+2) + \\ & + b_0^i u^0(k+1|k) + b_1^i u^0(k|k) + \cdots + b_{n_B}^i u(k-n_B+1) + d(k) \end{aligned} \tag{3.199}$$

$$y^0(k+2|k) = \sum_{i=1}^{r} \tilde{w}^i(k+1|k) y^{0i}(k+2|k)$$

where $u^0(k+1|k) = u^0(k|k) = u(k-1)$, while $\tilde{w}^i(k+1|k)$ are normalized activation levels of the rules (3.199). The above formulae can be written in a concise form

$$\begin{aligned} y^0(k+2|k) = & -a_1(k+1|k) y^0(k+1|k) - \ \cdots \ - a_{n_A}(k+1|k) y(k-n_A+2) + \\ & + b_0(k+1|k) u^0(k+1|k) + \cdots + b_{n_B}(k+1|k) u(k-n_B+1) + d(k) \end{aligned} \tag{3.200}$$

where

$$a_j(k+1|k) = \sum_{i=1}^{r} \tilde{w}^i(k+1|k) a_j^i, \ \ b_j(k+1|k) = \sum_{i=1}^{r} \tilde{w}^i(k+1|k) b_j^i$$

Predictions of the free outputs for next sampling instants $k+p$, $p = 3, ..., N$, can be performed in an analogous way. The general form of the prediction formula, for $p = 1$ and $p = 2$ yielding (3.198) and (3.200), is the following

$$\begin{aligned} y^0(k+p|k) = & - \sum_{j=1}^{\min\{n_A, p-1\}} a_j(k+p-1|k) y^0(k+p-j|k) + \\ & - \sum_{j=\min\{n_A,p-1\}+1}^{n_A} a_j(k+p-1|k) y(k+p-j) + \\ & + \sum_{j=0}^{\min\{n_B+1,p\}-1} b_j(k+p-1|k) u^0(k+p-1-j|k) + \\ & + \sum_{j=\min\{n_B+1,p\}}^{n_B} b_j(k+p-1|k) u(k+p-1-j) + d(k) \end{aligned} \tag{3.201}$$

where

$$a_j(k+p-1|k) = \sum_{i=1}^{r} \tilde{w}^i(k+p-1|k) a_j^i \tag{3.202}$$

$$b_j(k+p-1|k) = \sum_{i=1}^{r} \tilde{w}^i(k+p-1|k) b_j^i \tag{3.203}$$

and $a_j(k|k) = a_j(k)$, $b_j(k|k) = b_j(k)$, see (3.197), where $\tilde{w}^i(k+p-1|k)$ are normalized activation levels of the rules of the fuzzy model, evaluated at sampling instant $k$ for the future instant $k+p-1$, while $u^0(k+j|k) = u(k-1)$ for $j \geq 0$.

To calculate the forced component of the output trajectory in the prediction horizon, it is necessary to calculate the dynamic matrix. Knowing coefficients (3.197) of a linear (linearized) model at sampling instant $k$, we can evaluate elements of the step response of this model, $s_j(k)$, $j = 1, 2, \ldots$, directly from (3.106)

$$s_j(k) = - \sum_{i=1}^{\min\{j-1, n_A\}} a_i(k) s_{j-i}(k) + \sum_{i=0}^{\min\{j-1, n_B\}} b_i(k) \tag{3.204}$$

Dynamic matrix $\mathbf{M}(k)$ can be calculated according to (3.33), inserting values $s_j(k)$ obtained above in place of $s_j$, $j = N_1, ..., N$.

A nonlinear method for evaluation of the matrix $\mathbf{M}(k)$ remains only to be discussed, possible to be applied when generating this matrix for needs of optimization of the control input increments, see (3.158). As it was generally presented when discussing the construction of the MPC-NPL algorithms in the previous section, the subsequent lines of the matrix $\mathbf{M}(k)$ are adjusted to elements $y^0(k+p|k)$ of a free trajectory of the predicted outputs, generated successively. Thus, the matrix will have the following form (compare with (3.33))

$$\mathbf{M}(k) = \begin{bmatrix} s_{N_1}^{(N_1-1|k)} & \cdots & s_1^{(N_1-1|k)} & 0 & \cdots & 0 \\ s_{N_1+1}^{(N_1|k)} & \cdots & s_2^{(N_1|k)} & s_1^{(N_1|k)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{N_u}^{(N_u-1|k)} & \cdots & s_{N_u-N_1+1}^{(N_u-1|k)} & s_{N_u-N_1}^{(N_u-1|k)} & \cdots & s_1^{(N_u-1|k)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ s_N^{(N-1|k)} & \cdots & s_{N-N_1+1}^{(N-1|k)} & s_{N-N_1}^{(N-1|k)} & \cdots & s_{N-N_u+1}^{(N-1|k)} \end{bmatrix} \tag{3.205}$$

where $s_j^{(p|k)} = s_j(k+p|k)$ are elements of the step response of a linear model with coefficients $a_j(k+p|k)$ and $b_j(k+p|k)$, see (3.202) and (3.203).

Having calculated the free output trajectory $\mathcal{Y}^0(k) = [y^0(k+N_1|k) \cdots y^0(k+N|k)]^T$ and dynamic matrix $\mathbf{M}(k)$, necessary for evaluation of the forced trajectory $\Delta\mathcal{Y}(k) = \mathbf{M}(k)\Delta\mathcal{U}(k)$, it is possible to perform optimization of the control inputs solving a quadratic programming problem. This concludes the operation of the FGPC-NPL algorithm at sampling instant $k$.

In case of the MPC-NPL+ algorithm, subsequent iterations of the free output trajectory are calculated in the same way, only initial sequence of process control inputs $\mathcal{U}^j(k)$ should be used in place of $\mathcal{U}^0(k)$. The control input increments should be then calculated as a solution to the optimization problem (3.160).

*FDMC-NPL Algorithm*

For the FDMC-NPL algorithm the way of proceeding is analogous as in the FGPC-NPL case discussed above. Only different formulae will occur in consequents of the fuzzy rules, the ones which correspond to the process model based on step response coefficients, see Section 3.2.1.

A fuzzy model for nonlinear prediction of a free output trajectory will be written in the form of a set of $r$ rules with antecedents in a general form such as above in the FGPC algorithm, and with consequents in the form

$$y^{0i}(k+p|k) = y(0) + \sum_{j=p+1}^{k+p} s_j^i \triangle u(k+p-j) + d(k)$$

compare with (3.13), where $s_j^i$ denotes $j-$th element of the step response of a linear model occurring in the $i$-th rule consequent, $i = 1, .., r$. Disturbance $d(k)$ is assumed to be constant in the prediction horizon and equal to

$$d(k) = y(k) - \left[ y(0) + \sum_{j=1}^{k} s_j(k-1) \triangle u(k-j) \right]$$

where $s_j(k-1)$ are coefficients of the step response of the linear model used at sampling instant $k-1$. Combining the above two equations we obtain

$$y^{0i}(k+p|k) = y(k) + \sum_{j=1}^{D-1} [s_{j+p}^i - s_j(k-1)] \triangle u(k-j)$$

where $D$ denotes horizon of the step response dynamics, see Section 3.2.1. Therefore, the *nonlinear prediction of the free output trajectory*, performed at sampling instant $k$ for the prediction horizon $N$, is realized by *subsequent (recurrent)* application of the following rules, for $p = 1, 2, ..., N$,

$$\begin{aligned}
\mathrm{R}^i : \mathrm{IF} \quad & y^0(k+p-1|k) \text{ is } A_0^i \text{ and } \cdots \text{ and } y^0(k+1|k) \text{ is } A_{p-2}^i \\
& \text{and } y(k) \text{ is } A_{p-1}^i \text{ and } \cdots \text{ and } y(k+p-1-n_R) \text{ is } A_{n_R}^i \\
& \text{and } u^0(k+p-1|k) \text{ is } B_0^i \text{ and } \cdots \text{ and } u^0(k|k) \text{ is } B_{p-1}^i \\
& \text{and } u(k-1) \text{ is } B_p^i \text{ and } \cdots \text{ and } u(k+p-1-m_R) \text{ is } B_{m_R}^i \\
\text{THEN} \quad & y^{0i}(k+p|k) = y(k) + \sum_{j=1}^{D-1} [s_{j+p}^i - s_j(k-1)] \triangle u(k-j)
\end{aligned}$$

together with the conclusion of the fuzzy reasoning

$$y^0(k+p|k) = \sum_{i=1}^{r} \tilde{w}^i(k+p-1|k) y^{0i}(k+p|k)$$

where $\tilde{w}^i(k+p-1|k)$ are normalized activation levels of the rules at sampling instant $(k+p-1|k)$, while $u^0(k|k) = \cdots = u^0(k+N-1|k) = u(k-1)$ are components of the initial trajectory $\mathcal{U}^0(k)$ of predicted control inputs (3.156). Denoting

$$s_j(k+p|k) = \sum_{i=1}^{r} \tilde{w}^i(k+p|k) s_j^i, \quad j = 1, 2, ...$$

the last formula can be written in the following form

$$y^0(k+p|k) = y(k) + \sum_{j=1}^{D-1} [s_{j+p}(k+p-1|k) - s_j(k-1)] \triangle u(k-j) \quad (3.206)$$

It should be emphasized that calculation of the components of the nonlinear free output trajectory should be performed subsequently, recurrently – because for evaluation of the value $y^0(k+p|k)$ it is necessary to evaluate first levels of activation $\tilde{w}^i(k+p-1|k)$ dependent on previous elements of the free output trajectory, due to their presence in the antecedents of the fuzzy rules.

Since we know values of the coefficients $s_j(k)$ of the step response of the linear (linearized) model used at sampling instant $k$, hence we have directly the dynamic matrix $\mathbf{M}(k)$ needed for calculation of the forced output trajectory in the prediction horizon. We can also immediately calculate, if necessary, its nonlinear version (3.205).

**Example 3.11**
This is an example for a simple illustration of the design and analysis of a fuzzy predictive controller for a process model already considered in Example 2.6 in Chapter 2. The model is described by the following fuzzy rules

$$\mathrm{R}^1: \text{ IF } \ y(k) \text{ is } Y_1 \ \text{ THEN } \ y^1(k+1) = 0.7y(k) + 0.8u(k) \quad (3.207)$$

$$\mathrm{R}^2: \text{ IF } \ y(k) \text{ is } Y_2 \ \text{ THEN } \ y^2(k+1) = 0.3y(k) + 0.2u(k) \quad (3.208)$$

Fuzzy sets $Y_1$ and $Y_2$ are defined by sigmoidal membership functions

$$\mu_{Y_1}(y(k)) = \frac{1}{1+\exp(6y(k))}$$
$$\mu_{Y_2}(y(k)) = 1 - \mu_{Y_1}(y(k))$$

presented in Figures 2.23 (a). Linear models occurring in rule consequents (3.207) and (3.208) have indeed different gains and dynamics, which can be easily seen looking at their step responses presented in Fig. 2.23 (b).

The form (3.192) of the fuzzy model is in the case of our example problem the following

$$y(k+1) = \mu_{Y_1}(y(k))[0.7y(k) + 0.8u(k)] + \mu_{Y_2}(y(k))[0.3y(k) + 0.2u(k)]$$
$$= -a_1(k)y(k) + b_0(k)u(k) \tag{3.209}$$

where

$$-a_1(k) = 0.7\mu_{Y_1}(y(k)) + 0.3\mu_{Y_2}(y(k))$$
$$b_0(k) = 0.8\mu_{Y_1}(y(k)) + 0.2\mu_{Y_2}(y(k))$$

and activation levels of the fuzzy rules are given by the formulae

$$w^1(k) = \mu_{Y_1}(y(k))$$
$$w^2(k) = \mu_{Y_2}(y(k)) = 1 - \mu_{Y_1}(y(k))$$

The presented model is indeed nonlinear and it is not possible to design a linear controller which would operate well in the entire control domain. This fact was checked by designing linear GPC controllers for operating points corresponding to the output values $y = -1$, $y = +1$ and $y = 0$. The first two controllers operated correctly in neighborhoods of their operating points, but were unacceptable globally (instability or very slow operation in the more distant area). The third controller, designed for a middle operating point ($y = 0$) turned out to be slightly better, but still did not fulfil expectations – this is shown by trajectories of the output and input signals presented in Fig. 3.62.

For the presented process the following nonlinear controllers were designed and compared: NSL, NPL (also NPL+) and NO (with nonlinear optimization), assuming $N = 6$, $N_u = 3$, $N_1 = 1$, $\mathbf{\Psi}(p) = \mathbf{I}$, $\mathbf{\Lambda}(p) = \lambda\mathbf{I}$.

The *NSL algorithm* is evaluated at each sampling instant $k$ as a linear GPC algorithm for the linear (linearized) model

$$y(k) = -a_1(k)y(k-1) + b_0(k)u(k-1)$$

The design can be performed applying one of the methods presented in Section 3.3.1.

In the *NPL algorithm* we calculate a nonlinear prediction of the free output trajectory assuming control input increments equal to zero in the prediction horizon and applying the following formulae

$$y^0(k+1|k) = -a_1(k)y(k) + b_0(k)u(k-1) + d(k)$$

$$y^0(k+p|k) = -a_1(k+p-1)y^0(k+p-1|k) +$$
$$+b_0(k+p-1|k)u(k-1) + d(k), \qquad p = 2, ..., N$$

where

$$-a_1(k+p-1|k) = 0.7\mu_{Y_1}(y^0(k+p-1|k)) + 0.3\mu_{Y_2}(y^0(k+p-1|k))$$

$$b_0(k+p-1|k) = 0.8\mu_{Y_1}(y^0(k+p-1|k)) + 0.2\mu_{Y_2}(y^0(k+p-1|k)),$$
$$p = 2, ..., N$$

$$d(k) = y(k) - [-a_1(k-1)y(k-1) + b_0\,(k-1)\,u(k-1)] \tag{3.210}$$

The easiest way to calculate the dynamic matrix $\mathbf{M}(k)$, which is necessary to evaluate the forced output trajectory, is to calculate coefficients of the step response of a linearized model using (3.106).

Let us consider, for a moment, the following design method: first to calculate, off-line using (3.106), elements of step responses of linear models occurring in rule consequents (3.207), (3.208) and then, for each step response, corresponding dynamic matrices $\mathbf{M}^1$ and $\mathbf{M}^2$. Then, during the on-line operation of the algorithm, dynamic matrix $\mathbf{M}(k)$ is calculated at each sampling instant $k$ from the formula

$$\mathbf{M}(k) = \mu_{Y_1}(y(k))\mathbf{M}^1 + \mu_{Y_2}(y(k))\mathbf{M}^2$$

It should be pointed out that the above method *is not equivalent* to the one when the matrix $\mathbf{M}(k)$ is calculated on-line, at each sampling instant $k$, on the basis of a step response of a linearized model. It generally leads to

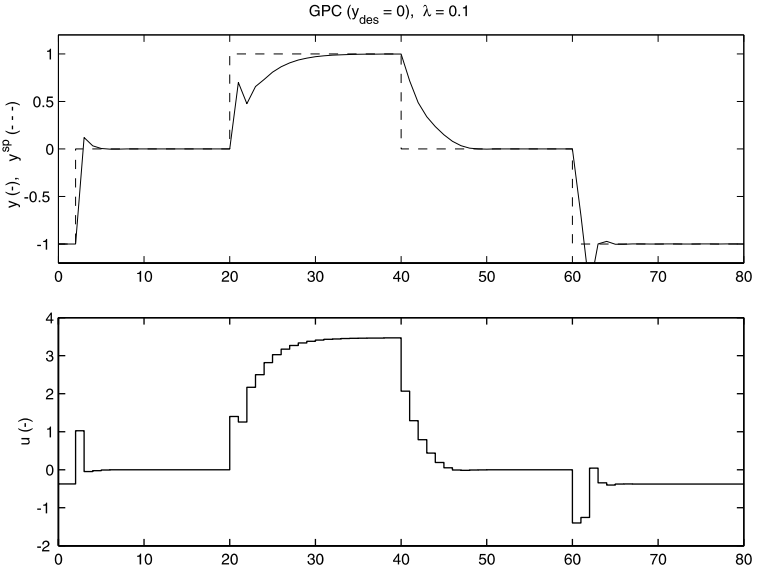


**Fig. 3.62.** Trajectories of the process input and output in the feedback control system with the GPC controller designed for $y = 0$

different results, though a predictive control algorithm constructed in such a way usually operates in a similar way.

Using the *NPL+ algorithm* we calculate subsequent free output trajectories $y^j(k+p|k)$, $p = 1, ..., N$ identically as the trajectory $y^0(k+p|k)$ above, only instead of the control input $u(k-1)$ it is necessary to use the value $u^j(k|k)$ (see description of the algorithm in Section 3.5.1). Moreover, it is necessary to remember about an appropriate modification of the controller optimization problem, to the form (3.160).

In an *NO type algorithm* at each step a nonlinear optimization is performed, elements of the trajectory of the predicted outputs over a prediction horizon are calculated using a nonlinear model (3.209), supplemented by a disturbance estimate given by (3.210). The procedure is recurrent, subsequent formulae look as follows:

$$y(k+1|k) = -a_1(k)y(k) + b_0(k)[u(k-1) + \triangle u(k|k)] + d(k)$$

$$y(k+2|k) = -a_1(k+1|k)y(k+1|k) + b_0(k+1|k)[u(k-1) + \triangle u(k|k) + \\ + \triangle u(k+1|k)] + d(k)$$

$$y(k+3|k) = -a_1(k+2|k)y(k+2|k) + b_0(k+2|k)[u(k-1) + \triangle u(k|k) + \\ + \triangle u(k+1|k) + \triangle u(k+2|k)] + d(k)$$

$$\vdots$$

$$y(k+6|k) = -a_1(k+5|k)y(k+5|k) + b_0(k+5|k)[u(k-1) + \triangle u(k|k) + \\ + \triangle u(k+1|k) + \triangle u(k+2|k)] + d(k)$$

In this case coefficients of the model are calculated based on full predicted trajectory of the outputs (and not only its free part), *i.e.*,

$$-a_1(k+p-1|k) = 0.7\mu_{Y_1}(y(k+p-1|k)) + 0.3\mu_{Y_2}(y(k+p-1|k))$$

$$b_0(k+p-1|k) = 0.8\mu_{Y_1}(y(k+p-1|k)) + 0.2\mu_{Y_2}(y(k+p-1|k)), \\ p = 2, ..., 6$$

Therefore, they depend on the vector of decision variables of the nonlinear optimization problem, $\Delta\mathcal{U}(k) = [\triangle u(k|k) \;\; \triangle u(k+1|k) \;\; \triangle u(k+2|k)]^T$.

Simulation studies were performed for the presented algorithms and different values of the coefficient $\lambda$. Figures from 3.63 to 3.66 present trajectories of the obtained process outputs and inputs, with $\lambda = 0.1$, by the following algorithms: linear GPC and nonlinear NSL, NPL, NPL+ and NO.

The FGPC-NSL algorithm turned out to be slightly better in the considered situation than the best linear one, namely the GPC algorithm designed for a medium point of nonlinearity ($y = 0$), mainly by decreasing the settling

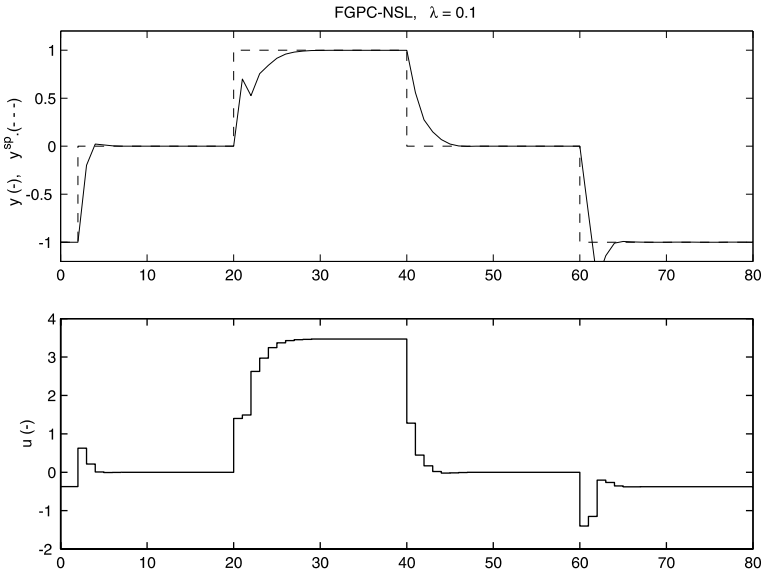


**Fig. 3.63.** Trajectories of the process input and output in the feedback control system with the FGPC-NSL controller

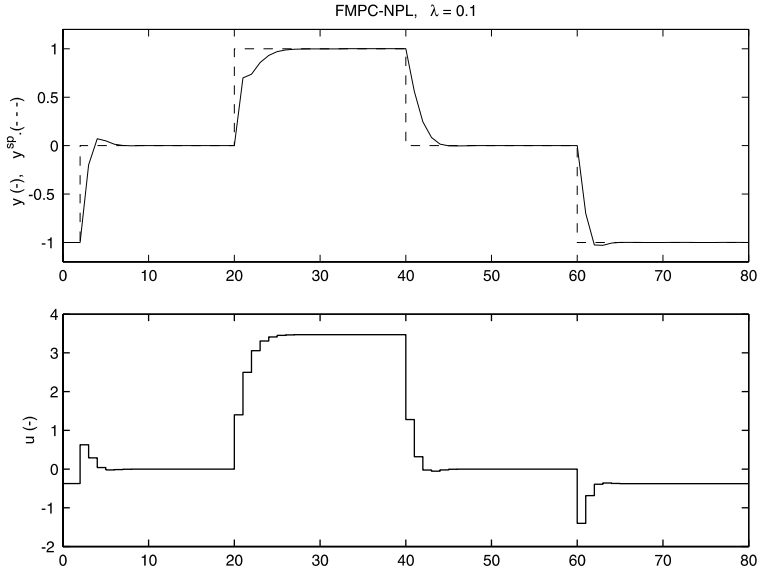


**Fig. 3.64.** Trajectories of the process input and output in the feedback control system with the FMPC-NPL controller

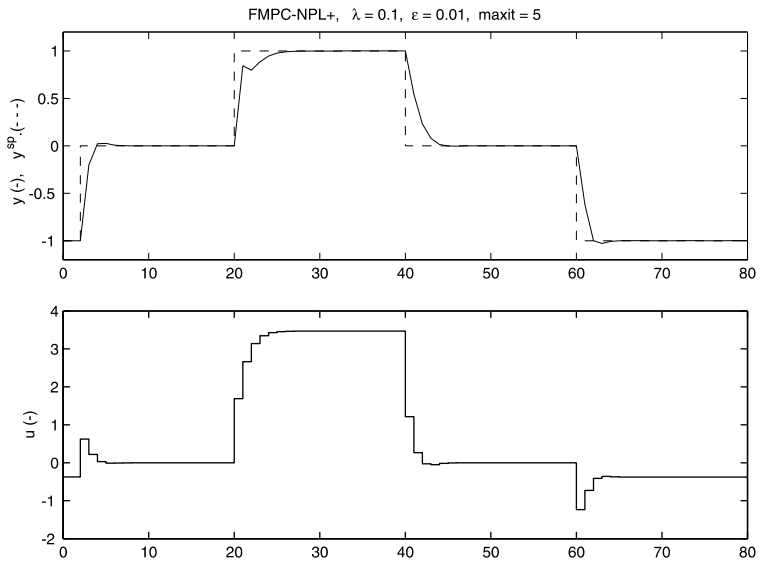


**Fig. 3.65.** Trajectories of the process input and output in the feedback control system with the FMPC-NPL+ controller

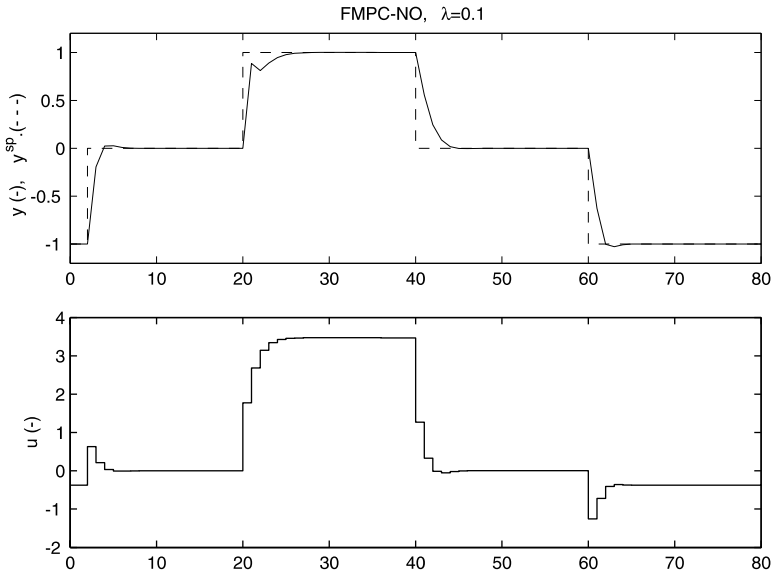


**Fig. 3.66.** Trajectories of the process input and output in the feedback control system with the FMPC-NO controller

time for a range of positive output values. Application of the FMPC-NPL algorithm (with a comparable complexity of calculations !) improved the control system properties, further reducing the settling time in the area $y \geq 0$ and practically eliminating a quite strong, disadvantageous overshoot present for negative values of $y$ (observed after a step change of the set-point value to $-1$). Applying the NPL+ algorithm has already caused an insignificant improvement.

It is worth noting that trajectories obtained with the NPL and NPL+ algorithms are similar to those obtained with a much more calculation-demanding FMPC-NO algorithm, with nonlinear optimization performed at each sampling instant (using a procedure from the *Optimization Toolbox* of the MATLAB® package). Moreover, in the case of the NPL+ and NO algorithms the trajectories are almost identical.

It is also interesting to compare the obtained results with the results for the control system with a PI fuzzy TS controller, presented in Example 2.6 in Section 2.5 □

### 3.5.8 Fuzzy MPC (FMPC) Explicit Unconstrained Algorithms

If only a linear process model is used and no constraints on values of process inputs and outputs are considered in the MPC optimization problem, then it has an unconstrained analytical solution, resulting in an unconstrained linear MPC control law. Such control laws were formulated for DMC, GPC and MPCS algorithms in Sections 3.2, 3.3 and 3.4, respectively. In the described situation it is possible to design FMPC algorithms in a different way than it was presented in previous sections, devoted to constrained numerical versions. The design procedure can than be as follows:

- First, explicit unconstrained predictive controllers are designed (as linear control laws), one for each fuzzy sub-domain corresponding to its associated fuzzy rule, having in its consequent a local linear process model, describing the process for control purposes in this sub-domain.
- Next, all the designed linear predictive control laws, supplemented by fuzzy reasoning, result in an overall nonlinear unconstrained fuzzy predictive control law.

The essence of the approach is an application of the general structure of the design of a nonlinear TS fuzzy controller, presented in Section 2.2. Since linear models were used for the design of local predictive controllers and the fuzzy control law is modified at each step of the algorithm, then the obtained nonlinear control algorithm is very similar to the FMPC-NSL type. However, it is not identical, as fuzzy reasoning applied to the local MPC control laws is not, in general, fully equivalent to the FMPC-NSL design procedure: fuzzy reasoning applied to local process models only to get a current linearization of the fuzzy model, with the following calculation (at each sampling instant) of the MPC control law for this model.

At the first stage, as always with fuzzy systems, *antecedents and consequents of inference rules of a TS fuzzy model* of the process are designed (see Chapter 2). This stage is based on defining the variables which describe operating points and on divisions of their variability ranges into fuzzy sets, such that in each of the created sub-domains the process can be well approximated by a local linear model. Next, for each sub-domain (*i.e.*, for each rule of the fuzzy model) a *local unconstrained linear predictive control law* is designed. Depending on the method of modeling and design, it will be a DMC, GPC or MPCS control law – if linear MPC algorithms presented earlier in this chapter are considered. The final nonlinear fuzzy MPC TS controller will be obtained as a combination of local controllers, by means of fuzzy reasoning.

Consider now the design of an explicit unconstrained FMPC controller taking as an example a *FDMC explicit controller* for a SISO process – with local linear DMC predictive control laws [89]. Assume that the first stage of the design was completed resulting in $r$ fuzzy sub-domains defined by rule antecedents of a TS fuzzy process model, where in each sub-domain a linear process model was chosen in the form of a step response of a finite length. It can be assumed, without loss of generality, that each of these step responses is of the same length, as all shorter responses can be appropriately lengthened. Next, assume that for each local model a DMC control law was designed. Assuming also that the set-point value for the controlled variable does not change over the prediction horizon, each of the local control laws will be given by (3.52). As a result, we obtain the explicit FDMC controller described by the following set of rules

$$\begin{aligned} \mathrm{R}_c^i : \mathrm{IF} \quad & y(k) \text{ is } A_0^i \text{ and } y(k-1) \text{ is } A_1^i \text{ and } \cdots \text{ and } y(k-n_R) \text{ is } A_{n_R}^i \\ & \text{and } u(k-1) \text{ is } B_1^i \text{ and } \cdots \text{ and } u(k-m_R) \text{ is } B_{m_R}^i \\ \mathrm{THEN} \quad & \triangle u^i(k) = (k^e)^i e(k) - \sum_{q=1}^{D-1} (k_q^u)^i \triangle u(k-q) \end{aligned} \tag{3.211}$$

where $i = 1, ..., r$, indexes fuzzy rules, and thus local DMC control laws, while $e(k) = y^{sp}(k) - y(k)$ denotes the control error. The controller output signal takes the form standard for the fuzzy TS structures

$$\triangle u(k) = \frac{\sum_{i=1}^{r} w^i(k) \triangle u^i(k)}{\sum_{l=1}^{r} w^l(k)} = \sum_{i=1}^{r} \tilde{w}^i(k) \triangle u^i(k) \tag{3.212}$$

where $\tilde{w}^i(k)$ are normalized activation levels of individual rules (3.211). Structure of the obtained FDMC explicit controller is presented in Fig. 3.67, where dotted lines visualize the fuzzy reasoning block and signals leading to it.

Of course, for applications where the process input signal can access the actuator's constraints on amplitude or rate of change, the structure in Fig. 3.67 should be modified by supplementing it with projection and anti-windup elements, in the same way as it was done for the DMC controller in Section 3.2.2, see Fig. 3.14.

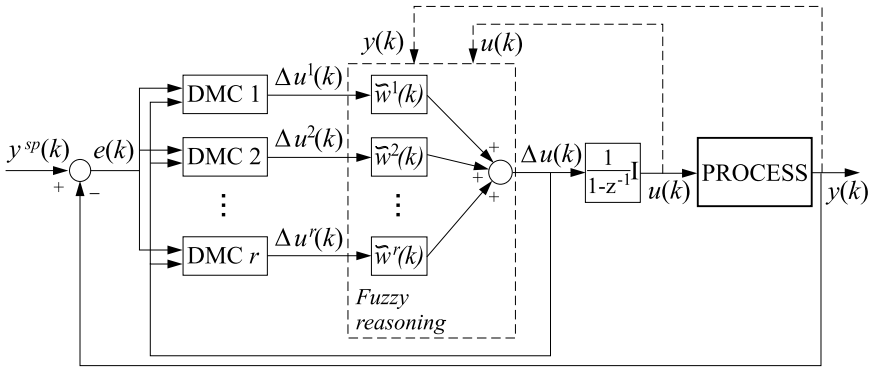


**Fig. 3.67.** Structure of the unconstrained explicit FDMC controller (the case with the set-point value constant in the prediction horizon)

Formulating the rules (3.211) we assumed a general form of their antecedents, the same as the one used in Section 2.2.2 devoted to discrete TS fuzzy output-feedback controllers. The presented FDMC explicit controller is a special case of the controller considered there, with rules (2.39), for $n_C = 1$, $m_D = D$ and

$$\begin{aligned} c_1^j &= (k^e)^j \\ d_1^j &= 1 - (k_1^u)^j \\ d_q^j &= (k_{q-1}^u)^j - (k_q^u)^j,\ q = 2, ..., D-1 \\ d_D^j &= (k_{D-1}^u)^j \end{aligned} \tag{3.213}$$

where $D$ is the length of step responses used during the design of local DMC controllers. Therefore, it is possible to use Corollary 2.6 formulated in Section 2.2.2, for stability analysis of the control system with the FDMC explicit controller [91, 92].

The analysis concerns a control system with a process modeled also by a TS fuzzy system, therefore a model of the controlled process in the same general form as in Section 2.2.2 will be assumed. Namely, in the form of a set of the following $ro$ rules (generally $ro \neq r$ and antecedents can also be different than in the controller rules)

$$\begin{aligned} \mathrm{R}_p^i : \mathrm{IF}\quad & y(k) \text{ is } A_0^i \text{ and } y(k-1) \text{ is } A_1^i \text{ and } \cdots \text{ and } y(k-n_O) \text{ is } A_{n_O}^i \\ & \text{and } u(k) \text{ is } B_0^i \text{ and } u(k-1) \text{ is } B_1^i \text{ and } \cdots \text{ and } u(k-m_O) \text{ is } B_{m_O}^i \\ \text{THEN}\quad & y^i(k+1) = a_1^i y(k) + a_2^i y(k-1) + \cdots + a_{n_A}^i y(k-n_A+1) \\ & + b_0^i u(k) + b_1^i u(k-1) + \cdots + b_{m_B}^i u(k-m_B) \end{aligned} \tag{3.214}$$

where $i = 1, ..., ro$. Activation levels of the rules (3.214) are denoted by $w_o^i(k)$ and their normalized values by $\tilde{w}_o^i(k)$. Therefore, the output of the process model is given by

$$y(k+1) = \frac{\sum_{i=1}^{ro} w_o^i(k)\, y^i(k+1)}{\sum_{l=1}^{ro} w_o^l(k)} = \sum_{i=1}^{ro} \tilde{w}_o^i(k)\, y^i(k+1) \tag{3.215}$$

Notice a slight difference when comparing notation used in (3.214) with the general form of the ARX type model applied in this section, compare with (3.105). Namely, in consequents of the rules (3.214) $m_B$ is used in place of $n_B$ – to be consistent with the notation used in Chapter 2, as we shall use the results presented in that chapter in the following considerations.

According to the Corollary 2.6, a sufficient stability condition for a control system with an explicit FDMC controller and the model (3.214), (3.215) is the existence of a positive definite matrix $\mathbf{P}$, such that for each matrix $\mathbf{A}_{ij}$ the equation $\mathbf{A}_{ij}^T \mathbf{P} \mathbf{A}_{ij} - \mathbf{P} < \mathbf{0}$ is fulfilled, for all pairs $(i, j)$, $i = 1, ..., ro$, $j = 1, ..., r$ except those for which always $w_o^i(k) w^j(k) = 0$. Matrices $\mathbf{A}_{ij}$ are given by (2.51), and in the considered case of our FDMC controller elements of the $(n+1)$-st row of these matrices result directly from (3.213), while elements of the first row are expressed by

$$\begin{aligned}
ac_1^{i,j} &= a_1^i - b_0^i (k^e)^j \\
ac_p^{i,j} &= a_p^i, \quad p = 2, ..., n_A \\
bd_1^{i,j} &= b_1^i + b_0^i [1 - (k_1^u)^j] \\
bd_q^{i,j} &= b_q^i + b_0^i [(k_{q-1}^u)^j - (k_q^u)^j], \quad q = 2, ..., D-1 \\
bd_D^{i,j} &= b_D^i + b_0^i (k_{D-1}^u)^j
\end{aligned} \tag{3.216}$$

where $n = n_A$ and $m = D$ was assumed, *i.e.*, $D \geq m_B$, as the number of elements in the step response is not smaller than the number of elements dependent on the process input variable in the ARX model (a model based directly on the step response is a limiting case). If $m_B < D$, then $b_q^i = 0$ for $m_B < q \leq D$ should be assumed in the above formulae.

In cases of identical rule antecedents both in the process model assumed for the design of the controller and in the process description for the control system simulation, we have $r = ro$, $w_o^i(k) = w^i(k)$, $i = 1, ..., r$. Then it is possible to lower the number of matrices $\mathbf{A}_{ij}$ and to use the formula in the form given in Corollary 2.7 for stability analysis, see Section 2.2.2.

A *FGPC fuzzy explicit controller*, with local unconstrained linear GPC control laws, can be designed in an analogous way. Instead of rules (3.211) we only have rules with consequents corresponding to (3.97)

$$\begin{aligned} \mathrm{R}_c^i : \text{IF} \quad & y(k) \text{ is } A_0^i \text{ and } y(k-1) \text{ is } A_1^i \text{ and } \cdots \text{ and } y(k-n_R) \text{ is } A_{n_R}^i \\ & \text{and } u(k-1) \text{ is } B_1^i \text{ and } \cdots \text{ and } u(k-m_R) \text{ is } B_{m_R}^i \\ \text{THEN} \quad & \triangle u^i(k) = (k^e)^i y^{sp}(k) - \sum_{q=0}^{n_{AR}} (k_q^y)^i y(k-q) - \sum_{q=1}^{m_{BR}} (k_q^u)^i \triangle u(k-q) \end{aligned} \tag{3.217}$$

$i = 1, ..., r$, where $n_{AR}$ and $m_{BR}$ correspond to parameters $n_A$ and $n_B$ of polynomials $A(z^{-1})$ and $B(z^{-1})$ of the process model (3.85) taken for the design of a GPC controller in Section 3.3.1. We have introduced here different symbols $n_{AR}$ and $m_{BR}$, because values $n_{AR}$ and $m_{BR}$ present in consequents of rules (3.217) (*i.e.*, in local GPC control laws) and values $n_A$ and $m_B$ in consequents of rules (3.214) representing the process in the control system structure can generally be different (a case of different process models used for controller design and for process representation in the feedback loop).

For stability analysis of a feedback control system with a FGPC controller and a process model in the form (3.214) we can also use Theorem 2.1, and precisely Corollary 2.6 presented in Section 2.2.2. However, notice that the formulae of local GPC controllers occurring in consequents of rules (3.217) are of a slightly different form than formulae of consequents of rules (2.39) in Section 2.2.2 – outputs of local controllers there depend on control errors and not directly on process outputs as in (3.217). This does not constitute an obstacle for the stability analysis, as the theses of both Theorem 2.1 and Corollary 2.6 do not depend on the signal $y^{sp}$. Therefore, it can be assumed, without loss of generality, that $y^{sp}(k) = y^{sp}(k-1) = \cdots = 0$. Then the GPC control laws occur to be special cases of control laws from consequents of the rules (2.39), for $n_C = n_{AR} + 1$, $m_D = m_{BR} + 1$ and

$$\begin{aligned} c_p^j &= (k_{p-1}^y)^j, \quad p = 1, ..., n_{AR} + 1 \\ d_1^j &= 1 - (k_1^u)^j \\ d_q^j &= (k_{q-1}^u)^j - (k_q^u)^j, \quad q = 2, ..., m_{BR} \\ d_{m_{BR}+1}^j &= (k_{m_{BR}}^u)^j \end{aligned} \tag{3.218}$$

Assuming $n = \max\{n_{AR} + 1, n_A\}$, $m = \max\{m_{BR} + 1, m_B\}$, we obtain the following formulae for elements of first rows of matrices $\mathbf{A}_{ij}$ (2.51) from Corollary 2.6:

$$\begin{aligned} ac_p^{i,j} &= a_p^i - b_0^i (k_{p-1}^y)^j, \quad p = 1, ..., n \\ bd_1^{i,j} &= b_1^i + b_0^i [1 - (k_1^u)^j] \\ bd_q^{i,j} &= b_q^i + b_0^i [(k_{q-1}^u)^j - (k_q^u)^j], \quad q = 2, ..., m \end{aligned} \tag{3.219}$$

In the above formulae:
– if $n_{AR} + 1 > n_A$, then $a_p^i = 0$ for $p > n_A$ should be assumed, and if $n_{AR} + 1 < n_A$ then $(k_{p-1}^y)^j = 0$ for $p > n_{AR} + 1$, and analogously

– if $m_{BR}+1 > m_B$ then $b_q^i = 0$ for $q > m_B$ should be assumed, and always $(k_q^u)^j = 0$ for $q > m_{BR}$.
Formulae for elements of $(n+1)$-st row of matrices $\mathbf{A}_{ij}$ follow directly from (3.218).

Design of *FMPCS fuzzy unconstrained explicit controllers*, using a fuzzy process model based on local process models in the form of linear state equations, can be performed in a quite analogous way. However, an additional design stage may be present here and should be commented on. Namely, there may be *estimated* state and disturbance variables in both antecedents and consequents (the latter being explicit MPCS control laws) of the fuzzy rules of the FMPCS explicit controller. These variables will be estimated using an observer, as discussed in Section 3.4.2 for linear processes. However, the process and its model are now nonlinear. Therefore, a nonlinear observer should be applied. As it was mentioned in Section 3.5.7, design of such observers is not as simple as in the linear case. It was also mentioned that it can be recommended to apply a nonlinear state observer in the form of a fuzzy TS system constructed on the basis of local linear observers [132], or an extended Kalman filter [77]. Discussing these questions is beyond the scope of this book.

**Example 3.12**
Let us consider a control system with the process from Example 3.11 and with the FGPC explicit unconstrained controller. The controller is obtained by designing local linear explicit unconstrained GPC control laws combined with fuzzy reasoning.

Assuming parameters established in Example 3.11, *i.e.*, $N = 6$, $N_u = 3$, $N = 1$ and $\lambda = 0.1$, for local linear models occurring in consequents of rules (3.207) and (3.208), the explicit GPC control laws being consequents of the following rules of the FGPC controller are obtained

$$\begin{aligned} &\mathrm{R}_c^1: \text{ IF } \ y(k) \text{ is } Y_1 \\ &\text{THEN } \ u^1(k) = 1.0012y^{sp}(k) - 1.7904y(k) + 0.7892y(k-1) \end{aligned} \tag{3.220}$$

$$\begin{aligned} &\mathrm{R}_c^2: \text{ IF } \ y(k) \text{ is } Y_2 \\ &\text{THEN } \ u^2(k) = 2.1948y^{sp}(k) - 2.9854y(k) + 0.7906y(k-1) \end{aligned} \tag{3.221}$$

Assuming the same membership functions as in Example 3.11 and denoting by $w^1(k)$ and $w^2(k)$ their activation levels (it is not necessary to normalize these values as from the construction $w^1(k) + w^2(k) = 1$), the FGPC control law can be written in the following form

$$\begin{aligned} u(k) &= w^1(k)u^1(k) + w^2(k)u^2(k) \\ &= [1.0012w^1(k) + 2.1948w^2(k)]y^{sp}(k) - [1.7904w^1(k) + \\ &+ 2.9854w^2(k)]y(k) + [0.7892w^1(k) + 0.7906w^2(k)]y(k-1) \end{aligned}$$

Trajectories obtained in the feedback control system with the designed controller are very similar to those shown in Figure 3.63 in Example 3.11, where a FGPC-NSL controller in a situation without activity of constraints was presented. Therefore, they will not be presented here.

For stability analysis of the feedback control system with the process model given by rules (3.207), (3.208) and the designed explicit FGPC controller the Corollary 2.7 with $\mathbf{Q} = \mathbf{0}$ can be used. In order to do this it is necessary to evaluate matrices $\mathbf{A}_{ij}$ (2.51). In the considered example we have $n = 2$ and $m = 1$, thus matrices $A_{ij}$ are of dimension 3, of a general form

$$\mathbf{A}_{ij} = \begin{bmatrix} ac_1^{ij} & ac_2^{ij} & bd_1^{ij} \\ 1 & 0 & 0 \\ -c_1^j & -c_2^j & d_1^j \end{bmatrix}, \quad i, j = 1, 2$$

with elements given by (3.218) and (3.219), where non-zero values of parameters of the local process models and controllers are

$$\begin{aligned} a_1^1 &= 0.7, \;\; b_0^1 = 0.8, \quad (k_1^y)^1 = 1.7904, \;\; (k_2^y)^1 = -0.7892 \\ a_1^2 &= 0.3, \;\; b_0^2 = 0.2, \quad (k_1^y)^2 = 2.9854, \;\; (k_2^y)^2 = -0.7906 \end{aligned}$$

According to the thesis of Corollary 2.7, it is enough to consider matrices $\mathbf{A}_{11}$, $\mathbf{A}_{22}$ and $\overline{\mathbf{A}_{12}} = (\mathbf{A}_{12} + \mathbf{A}_{21})/2$, which in the analyzed case are

$$\mathbf{A}_{11} = \begin{bmatrix} -0.7496 & 0.6361 & 0.8000 \\ 1 & 0 & 0 \\ -1.8120 & 0.7951 & 1 \end{bmatrix}$$

$$\mathbf{A}_{22} = \begin{bmatrix} -0.3169 & 0.1626 & 0.2000 \\ 1 & 0 & 0 \\ -3.0846 & 0.8131 & 1 \end{bmatrix}$$

$$\overline{\mathbf{A}_{12}} = \begin{bmatrix} -0.9150 & 0.4048 & 0.5000 \\ 1 & 0 & 0 \\ -2.4483 & 0.8041 & 1 \end{bmatrix}$$

Using the MATLAB® package, it was checked that the obtained system of linear matrix inequalities

$$\begin{aligned} (\mathbf{A}_{11})^T \mathbf{P} \mathbf{A}_{11} - \mathbf{P} &< 0 \\ (\mathbf{A}_{22})^T \mathbf{P} \mathbf{A}_{22} - \mathbf{P} &< 0 \\ (\overline{\mathbf{A}_{12}})^T \mathbf{P} \overline{\mathbf{A}_{12}} - \mathbf{P} &< 0 \\ -\mathbf{P} &< 0 \end{aligned}$$

has a solution in the form of the following symmetric matrix $\mathbf{P}$

$$\mathbf{P} = \begin{bmatrix} 12.1556 & -5.8906 & -5.6196 \\ -5.8906 & 4.2805 & 3.4626 \\ -5.6196 & 3.4626 & 3.4601 \end{bmatrix}$$

Therefore, due to Corollary 2.7, sufficient stability conditions for the considered nonlinear control system are satisfied. □

In [91, 92, 87] examples of design and stability analysis of explicit unconstrained FDMC controllers can be found, in particular for the process considered in the last example, for concentration control in an ethylene distillation column and for MIMO control of a vaporizer.

## 3.6 Stability, Constraint Handling, Parameter Tuning

### 3.6.1 Stability of MPC Algorithms

It is an interesting phenomenon that predictive control algorithms with a finite receding horizon have been successfully applied in industrial practice for several years, before theoretical results concerning their stability analysis appeared. Moreover, this fact did not constitute an obstacle in their development, in production of commercial products with good properties and practical popularity. Analysis based on computer simulations and experience gathered in subsequent applications turned out to be sufficient. Today, we know a lot more about conditions of stability of the MPC algorithms, and first of all about the methods of their formulation or modifications which guarantee stability. This knowledge, however, is used not only for designing algorithms with a guaranteed nominal stability, but first of all it helps to achieve a better understanding of the mechanism of operation of the predictive control and provides guidelines for a better justified selection of elements of their structure and parameter values. In face of the complexity of the MPC algorithms, especially in a situation of multiple constraints both on process input and output variables, the problem of stability is important, but it is not the only one. Ensuring non-emptiness of the set of feasible solutions at each sampling instant is just as important. Heuristic approaches and analysis based on computer simulations still play an important role.

Basic, selected results concerning stability of the MPC algorithms, especially those with linear models, will be presented in this section. The presentation has no pretence to be complete or rigorous mathematically – this would exceed the scope of this book. If a reader is interested in more extensive studies on the subject, we recommend an excellent review article by Mayne *et al.* [94] and publications listed in its bibliography, see also [98, 1, 82, 123, 115]. The purpose of considerations of this section is, first of all, a presentation of basic mechanisms which influence stability and resulting indications which explain both directions of modifications of the controller structure and guidelines how to select values of the controller parameters.

The question of stability has been touched on already in this chapter – in cases of explicit, unconstrained formulations of MPC algorithms as explicit feedback control laws, which were possible to be obtained when the constraints

on input and output variables were not considered. For controllers with linear process models, as DMC, GPC or MPCS algorithms, the control law is a linear output-feedback or state-feedback, see Sections 3.2, 3.3 and 3.4. Stability of a control system with such a controller can be analyzed using methods known in linear control theory. Using a specific structure of nonlinear TS fuzzy models, it was also possible to derive an analytical way of formulating sufficient stability conditions of nonlinear predictive controllers with such models, of course also in the case of a design without constraints, see Section 3.5.8.

Let us emphasize that considering a situation without constraints constitutes a recommended stage of the design procedure of a predictive controller, also when its operation in the presence of constraints is foreseen. That is because working points may be located inside the controller admissible set, and because we have a better understanding of stability of the feedback control loop in a situation without constraints. However, in cases with constraints only on control input values, an explicit controller implemented in the correct feedback and anti-windup structure can operate quite well, see Section 3.2. Certainly, this does not change the fact that the main problem and challenge is the question of stability of the MPC algorithms in numerical versions, with the optimization problem with constraints solved numerically at every sampling instant.

**Basic Mechanisms Ensuring Stability**

Two elements in the formulation of an MPC algorithm have a key significance for the stability of the resulting control system:

- *A constraint on terminal state*, *i.e.*, a constraint on state variables at the end of the prediction horizon,
- *Length of the prediction horizon* $N$.

Enforcing a certain state value at the end of the prediction horizon is sufficient to guarantee stability even in the general nonlinear case – but under assumption of feasibility only, *i.e.*, when the set of admissible control input values (decision variables in the controller optimization problem) is nonempty at each step of the predictive algorithm. The proof is relatively easy, if we apply reasoning based on the Lyapunov function, as it was shown in the original papers by Chan and Shaw [24], for a control system with continuous time, see also [104], or by Keerthi and Gilbert [63] for a system with discrete time.

Namely, let us consider a process described by a nonlinear difference equation

$$x(k+1) = f(x(k), u(k)) \tag{3.222}$$

assuming that the state $(x, u) = (0, 0)$ is its equilibrium point. For simplicity, assume also that prediction and control horizons are equal, *i.e.*, $N_u = N$, and denote, as before in this book, by $x(k+p|k)$ the state predicted at sampling instant $k$ for the future instant $k+p$, using the above model and control input values $u(k+p-1|k)$, $p = 1, ..., N$. Next, consider a cost function in the form

$$J(k) = \sum_{p=1}^{N} h(x(k+p|k), u(k+p-1|k)) \tag{3.223}$$

where $h(x,u) \geq 0$ and $h(x,u) = 0$ if and only if $x = 0$ and $u = 0$. Additionally, let us take into account the following constraints

$$u(k+p|k) \in U, \quad p = 0, 1, ..., N-1 \tag{3.224a}$$
$$x(k+p|k) \in X, \quad p = 1, 2, ..., N-1 \tag{3.224b}$$
$$x(k+N|k) = 0 \tag{3.224c}$$

The last constraint is precisely the mentioned *terminal state constraint*. To calculate next control inputs we shall use a general rule of predictive control, *i.e.*, at each sampling instant the cost function (3.223) is optimized under all mentioned constraints, (3.222) and (3.224a)-(3.224c), while for controlling the process only the first element $\hat{u}(k) = \hat{u}(k|k)$ of the calculated optimal vector $[\hat{u}(k|k) \ \cdots \ \hat{u}(k+N-1|k)]^T$ is taken.

Denote by $\hat{J}(k)$ the optimal value of the cost function at sampling instant $k$ (under the constraints formulated above), similarly, by $\hat{J}(k+1)$ optimal value of the cost function at the next sampling instant $k+1$. Denote by $\hat{x}(k+p|k)$, $p = 1, 2, ..., N$ the optimal state trajectory evaluated at sampling instant $k$, corresponding to the optimal value $\hat{J}(k)$. One can then write

$$\begin{aligned}
\hat{J}(k+1) &= \min_{u(k+1),\ldots,u(k+N)} \sum_{p=1}^{N} h(x(k+1+p|k+1), u(k+p)) \\
&= \min_{u(k+1),\ldots,u(k+N)} \left\{ \sum_{p=1}^{N} h(x(k+p|k+1), u(k+p-1)) + \right. \\
&\left. - \ h(x(k+1|k+1), u(k)) + h(x(k+1+N|k+1), u(k+N)) \right\}
\end{aligned}$$

$$\begin{aligned}
&\leq \hat{J}(k) - h(\hat{x}(k+1|k), \hat{u}(k|k)) + \\
&\qquad + \min_{u(k+N)} h(x(k+1+N|k+1), u(k+N))
\end{aligned} \tag{3.225}$$

because the vector

$$[\hat{u}(k+1|k) \ \cdots \ \hat{u}(k+N-1|k) \ \ u^*(k+N|k+1)]^T$$

cannot be better at sampling instant $k+1$ than the optimal vector for this instant ($N-1$ first elements of the above vector are components of the optimal vector for the previous sampling instant $k$, while $u^*(k+N|k+1)$ is the value which minimizes the last component of inequality (3.225)). As the constraint $x(k+N|k) = 0$ is satisfied after optimization at sampling instant $k$, and the point $(0,0)$ is the equilibrium point, then $u^*(k+N|k+1) = 0$, as then $x(k+1+N|k+1) = 0$ and the last component on the right-hand side of

the inequality (3.225) takes the smallest possible value equal to zero. Thus we obtain

$$\hat{J}(k+1) - \hat{J}(k) \leq -h(\hat{x}(k+1|k), \hat{u}(k|k)) \leq 0$$

and if we take function $\hat{J}(k)$ as a Lyapunov function, then there follows from the Lyapunov theorem that the control system with the considered predictive controller is stable.

The possibility to prove the above very strong result follows, first of all, from the assumption that the algorithm feasible set defined by the set of constraints (3.224) is not empty at each sampling instant. Moreover, there is an implicit assumption that after each optimization (at each sampling instant) a global minimum of the cost function $J(k)$ is obtained – which for linear models and typical quadratic cost functions is a natural and always fulfilled assumption, while for nonlinear models it is difficult to be guaranteed. No doubt, the key assumption is that of a non-emptiness of the feasible set, which is, generally, strongly constrained due to the additional requirement $x(k+N|k) = 0$.

Using the idea of enforcing the terminal state to obtain stability, Clarke and Scattolini proposed a version of an explicit, unconstrained GPC algorithm with a constraint of this type added, called the CRHPC (*Constrained Receding-Horizon Predictive Control*) [28] algorithm. In [126] this formulation was extended to the case of a numerical algorithm, with inequality constraints on process inputs. The formulation of the optimization problem of the CRHPC algorithm is an extension of the optimization problem of the GPC algorithm (see Section 3.3), by adding additional constraints

$$y(k+N+j|k) = y^{sp}(k+N), \quad j = 1, ..., m \tag{3.226}$$

In [126] stability of the CRHPC algorithm was shown, for $N_u = N - \tau + 1$ and $N_u \geq m$, for $m = \max\{n_A, n_B - \tau + 1\} + 1$, where $n_A$ and $n_B$ are degrees of polynomials $A(z^{-1})$ and $B(z^{-1})$ in the process model used in the GPC algorithm, see Section 3.3, while $\tau$ is a delay in the process control path. Of course, it is also necessary to have non-emptiness of the feasible set as an assumption – *i.e.*, the assumption that inequality constraints together with the equality constraints (3.226) can be satisfied at each sampling instant. Let us notice that prediction of the output values is in fact performed over a horizon of the length $N + m$. The constraints should be considered until the instant $k + N + m$, even though in the cost function summing is only up to $N$. Additional equality constraint is forced not at one, but at $m$ subsequent sampling instants in order to ensure stabilization of the entire process state. And finally, observe that a long control horizon $N_u$ is used, as this minimizes the risk that the constraint set given by (3.226) becomes empty. The authors recommend the CRHPC algorithm for difficult processes, when applications of GPC are troublesome or there are troubles with the stability.

Consider now a standard GPC algorithm, but with the control horizon equal to the prediction horizon, $N_u = N$. Observe now, that if the influence

of coefficients of the cost function, expressed by matrices $\mathbf{\Psi}(p)$ and $\mathbf{\Lambda}(p)$, does not decrease along with an increase in $p$ (*e.g.*, $\mathbf{\Psi}(p) = \psi\mathbf{I}$ and $\mathbf{\Lambda}(p) = \lambda\mathbf{I}$, are not dependent on $p$) the prediction horizon is sufficiently long and inequality constraints do not contradict achieving the desired, constant set-point, then process outputs should stabilize on the set-point values at the end of the prediction horizon , *i.e.*, $\hat{y}(k+N+j|k) = y^{sp}(k+N),\ j = 1, ..., m$ should be true, with a sufficient accuracy. Thus, we obtain a situation enforced in the CPHPC algorithm by the additional constraint (3.226), without imposing this constraint, but only extending the length of the horizons. Thus it can be concluded that a hypothesis that extending the length of the horizons has stabilizing properties, may be true.

For infinite prediction and control horizons ($N = N_u = \infty$) nominal stability of control systems follows directly from the general rule of operation of predictive controllers. For a proof, it is enough to consider the Bellman optimality principle (in brief, for additive objective functions: each terminal interval of an optimal trajectory is also, starting from its initial point, an optimal trajectory). It follows from this principle that, if the external conditions (trajectory of set-point values) remain unchanged and there are no modeling errors, then the system with a predictive controller will move exactly along the optimal trajectory $[\hat{u}(0|0)\ \hat{u}(1|0)\ \hat{u}(2|0)\ ...]^T$ calculated at the initial sampling instant, *i.e.*, $\hat{u}(k|k) = \hat{u}(k|0)$, $k = 0, 1, 2, ....$ Therefore, the control system will be stable. This situation is shown in Fig. 3.68 (a). To compare, Fig. 3.68 (b) presents possible trajectories occurring when a finite prediction horizon is used.

In practical applications of predictive control algorithms, the length of prediction and control horizons, and particularly the length of the control horizon, strongly influences the computation time necessary to calculate the control input trajectory at each sample. The longer the horizon, the longer the calculation time, and too long a calculation time required may be critical in practical applications. Therefore, to obtain a guaranteed stability, with a sufficient safety margin, by means of significant lengthening of the horizons can not be acceptable as a general way of proceeding. On the other hand, it is precisely the case of shorter horizons when the application of an alternative stabilizing approach, by enforcing the terminal state, may easily result in impossibility to reach this state, due to the resulting emptiness of the feasible set. Therefore, it is also generally not acceptable as a too restrictive way of proceeding. Fortunately, it is easy to show that the key factor for better stability is the *infinite prediction horizon*, not the control horizon.

Consider the case of a stable, nonlinear process (3.222) which should be regulated to its zero equilibrium point, assuming the output equation $y(k) = g(x(k))$. Let the cost function be defined on an infinite horizon, with matrices of weighting coefficients not depending on time, $\mathbf{\Psi}(p) = \mathbf{\Psi}$ and $\mathbf{\Lambda}(p) = \mathbf{\Lambda}$, and with both control input increments and control input values taken into account [100]

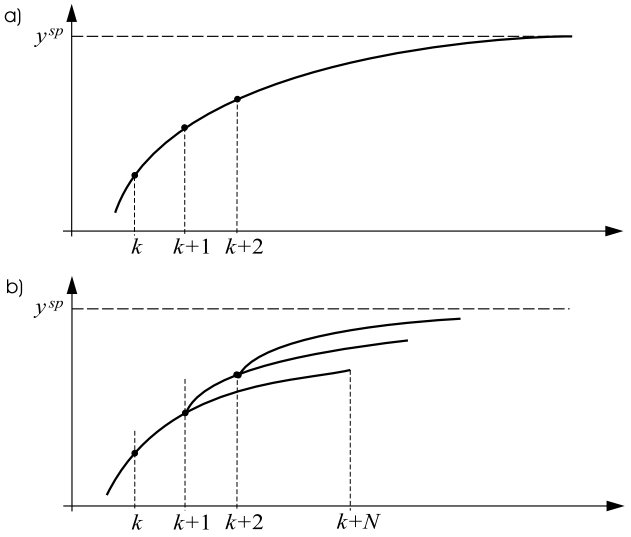


**Fig. 3.68.** Example process output trajectories when using predictive control with: (a) infinite prediction and control horizons; (b) finite prediction and control horizons equal to $N$ (assuming ideal process model)

$$J(k) = \sum_{p=1}^{\infty} \left( \|y(k+p|k)\|^2_{\mathbf{\Psi}} + \|\triangle u(k+p-1|k)\|^2_{\mathbf{\Lambda}} + \|u(k+p-1|k)\|^2_{\mathbf{R}} \right) \tag{3.227}$$

where matrices $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ are positive semidefinite, matrix $\mathbf{R}$ is positive definite. Assume also a *finite control horizon* of length $N_u \geq 1$, *i.e.*,

$$\triangle u(k+p|k) = 0 \quad \text{for } p \geq N_u, \ k = 0, 1, ... \tag{3.228}$$

As previously, denote by $\hat{J}(k)$ optimal value of the cost function obtained by its optimization with respect to the vector $[\triangle u(k|k) \ \triangle u(k+1|k) \ \cdots \ \triangle u(k+N_u-1|k)]^T$, under additional constraints on state or control input, assuming that the feasible set is nonempty. Then we have

$$\hat{J}(k) = \sum_{p=1}^{\infty} \|\hat{y}(k+p|k)\|^2_{\mathbf{\Psi}} + \sum_{p=0}^{N_u-1} \left( \|\triangle \hat{u}(k+p|k)\|^2_{\mathbf{\Lambda}} + \|\hat{u}(k+p|k)\|^2_{\mathbf{R}} \right) \tag{3.229}$$

because $\hat{u}(k+N_u-1|k) = 0$ must be true – otherwise the value of $J(k)$ would be infinite. In the above formula $\hat{u}(k+p-1|k)$, $p = 1, ..., N_u$ denote optimal control input values calculated at sampling instant $k$, and $\hat{y}(k+p|k)$, $p = 1, 2, ...$ the corresponding output values. As in the previous case of an additional terminal state constraint, taking $\hat{J}(k)$ as a candidate for a Lyapunov function, we can write

$$\hat{J}(k+1) = \sum_{p=1}^{\infty} \|\hat{y}(k+1+p|k+1)\|^2_{\mathbf{\Psi}} + \\ + \sum_{p=0}^{N_u-1} \left( \|\triangle\hat{u}(k+1+p|k+1)\|^2_{\mathbf{\Lambda}} + \|\hat{u}(k+1+p|k+1)\|^2_{\mathbf{R}} \right) \\ \leq \hat{J}(k) - \|\hat{y}(k+1|k)\|^2_{\mathbf{\Psi}} - \|\triangle\hat{u}(k|k)\|^2_{\mathbf{\Lambda}} - \|\hat{u}(k|k)\|^2_{\mathbf{R}}$$

because optimal control input values evaluated at sampling instant $k$ cannot give, at the next sampling instant $(k+1)$, better result than optimal ones evaluated for that instant. Thus we obtain $\hat{J}(k+1) - \hat{J}(k) \leq 0$, and because the sequence of values $\hat{J}(k)$ is constrained from the bottom, then it is convergent. It follows directly from this reasoning that

$$\|\hat{y}(k+1|k)\|^2_{\mathbf{\Psi}} + \|\triangle\hat{u}(k|k)\|^2_{\mathbf{\Lambda}} + \|\hat{u}(k|k)\|^2_{\mathbf{R}} \to 0 \tag{3.230}$$

and positive definitness of $\mathbf{R}$ implies that $\hat{u}(k|k) = u(k) \to 0$. The assumption of process stability implies then that $x(k) \to 0$, *i.e.*, implies *stability of the predictive control system*. Recall also that assumptions: that the feasible set in the optimization problem is nonempty, and that global minimum of this problem is found, must be satisfied at each sampling instant.

The result proven suggests that increasing the length of a prediction horizon while keeping the control horizon finite stabilizes the operation of the predictive control system. Certainly, to guarantee stability in this way a practical problem of how to effectively calculate the value of the cost function over an infinite horizon should be solved. The solution to this problem, for linear process models, was proposed in [117, 100].

### Control with a Linear Model and an Infinite Prediction Horizon

A starting point for the considerations is a description of the process by a set of linear, stable state equations

$$x(k+1) = \mathbf{A}x(k) + \mathbf{B}u(k)$$
$$y(k) = \mathbf{C}x(k)$$

and stating the control problem as a regulation to the zero equilibrium point.

Before we pass to the problem how to calculate the cost function on an infinite horizon, observe that in typical formulations of MPC algorithms with input-output type linear process models, as in the case of DMC or GPC, there are usually no components in the cost function representing squares of control input values, *i.e.*, there is $\mathbf{R} = \mathbf{0}$. However, the reasoning presented at the previous point is still correct for $\mathbf{R} = \mathbf{0}$, if only $\mathbf{\Psi} > \mathbf{0}$ and $y(k) \to 0$ implies $x(k) \to 0$, *i.e.*, there are not process modes unobservable by the controlled outputs Remember that nominal stability is considered, *i.e.*, the case with a precise model and without disturbances, then with $\mathbf{\Psi} > \mathbf{0}$ it follows from

(3.230) that $\hat{y}(k+1|k) = y(k+1) \to 0$. Let us comment also that in algorithms using input-output process descriptions only, such as DMC or GPC, the considered process model represents only a controllable and observable part. Therefore, stability is considered in the sense of a stable behavior of the process outputs, *i.e.*, for controllable and observable modes only.

To give an effective method of calculating the first (infinite) sum in (3.227), it will be divided into two components

$$\sum_{p=1}^{\infty} \|y(k+p|k)\|_{\mathbf{\Psi}}^2 = \sum_{p=1}^{N_u-1} \|y(k+p|k)\|_{\mathbf{\Psi}}^2 + \sum_{p=N_u}^{\infty} \|y(k+p|k)\|_{\mathbf{\Psi}}^2$$

Then the second sum can be presented in the form

$$\begin{aligned}\sum_{p=N_u}^{\infty} \|y(k+p|k)\|_{\mathbf{\Psi}}^2 &= \sum_{p=N_u}^{\infty} x(k+p|k)^T \mathbf{C}^T \mathbf{\Psi} \mathbf{C} x(k+p|k) \\ &= x(k+N_u|k)^T \left[ \sum_{i=0}^{\infty} (\mathbf{A}^T)^i \mathbf{C}^T \mathbf{\Psi} \mathbf{C} \mathbf{A}^i \right] x(k+N_u|k)\end{aligned}$$

because, as we have shown above, there must be $\hat{u}(k+p-1|k) = 0$ for $p \geq N_u$. Thus

$$\begin{aligned} y(k+N_u|k) &= \mathbf{C} x(k+N_u|k) \\ y(k+N_u+1|k) &= \mathbf{C}\mathbf{A} x(k+N_u|k) \\ y(k+N_u+2|k) &= \mathbf{C}\mathbf{A}^2 x(k+N_u|k), \quad \textit{etc.} \end{aligned} \tag{3.231}$$

The matrix

$$\mathbf{\Psi}^{\infty} = \sum_{i=0}^{\infty} (\mathbf{A}^T)^i \mathbf{C}^T \mathbf{\Psi} \mathbf{C} \mathbf{A}^i$$

exists for a matrix $\mathbf{A}$ of a stable process and can be obtained from a solution of a discrete Lyapunov equation

$$\mathbf{A}^T \mathbf{\Psi}^{\infty} \mathbf{A} + \mathbf{C}^T \mathbf{\Psi} \mathbf{C} = \mathbf{\Psi}^{\infty}$$

because

$$\mathbf{\Psi}^{\infty} = \mathbf{C}^T \mathbf{\Psi} \mathbf{C} + \sum_{i=1}^{\infty} (\mathbf{A}^T)^i \mathbf{C}^T \mathbf{\Psi} \mathbf{C} \mathbf{A}^i = \mathbf{C}^T \mathbf{\Psi} \mathbf{C} + \mathbf{A}^T \mathbf{\Psi}^{\infty} \mathbf{A}$$

In conclusion, the optimization problem of a predictive controller with an infinite prediction horizon can be presented in the following form of a problem with a finite horizon and a penalty component for the terminal state (at sampling instant $k+N_u$), defined by the matrix $\mathbf{\Psi}^{\infty}$,

$$J(k) = x(k+N_u|k)^T \mathbf{\Psi}^\infty x(k+N_u|k) + \sum_{p=1}^{N_u-1} \|y(k+p|k)\|^2_{\mathbf{\Psi}} + \\ + \sum_{p=0}^{N_u-1} \left( \|\triangle u(k+p|k)\|^2_{\mathbf{\Lambda}} + \|u(k+p|k)\|^2_{\mathbf{R}} \right)$$

If there are no state constraints (implied by, *e.g.*, output constraints) which are to be satisfied on an infinite horizon, then it should be accepted that $N_u$ is the prediction horizon $N$, since there is no need to evaluate state values $x(k+p|k)$ for $p > N_u$. If, however, such constraints exist, then they should be considered in the formulation of the optimization problem, over a horizon of length $N \geq N_u$, where $N$ is the smallest natural number such that satisfying constraints on the prediction horizon $N$ will ensure their satisfaction for all the subsequent sampling instants $k+p > k+N$. It was shown in [117] that such a finite number exists. In the presented situation the case $N > N_u$ can easily happen, especially for smaller values of $N_u$. Of course, an effective prediction horizon which requires calculation of predicted outputs, is then $N$.

In practice, set-point values for outputs are usually non-zero, corresponding to non-zero equilibrium points. It is then necessary to use a cost function in the form

$$J(k) = [x(k+N_u|k) - x^{ss}]^T \mathbf{\Psi}^\infty [x(k+N_u|k) - x^{ss}] + \\ + \sum_{p=1}^{N_u-1} \|y(k+p|k) - y^{sp}\|^2_{\mathbf{\Psi}} + \\ + \sum_{p=0}^{N_u-1} \left( \|\triangle u(k+p|k)\|^2_{\mathbf{\Lambda}} + \|u(k+p|k) - u^{ss}\|^2_{\mathbf{R}} \right) \qquad (3.232)$$

where $u^{ss}$ and $x^{ss}$ are process input and state values which correspond to the set-point value $y^{sp}$ in a steady-state [100]. These values should be calculated from equations of process statics,

$$0 = \mathbf{A}x^{ss} + \mathbf{B}u^{ss}$$
$$y^{sp} = \mathbf{C}x^{ss}$$

Until now only stable processes have been considered. For *an unstable process* described by state equations it is necessary to perform a Jordan decomposition of the state matrix $\mathbf{A}$ of the process model, into stable and unstable parts. Further, the controller optimization problem must be supplemented by an equality constraint enforcing zeroing of unstable process modes at the end of the prediction horizon $N$. Then, the infinite sum can be calculated only for stable modes, in an identical way as it is done for a stable process (unstable modes remain zero for $k+p > k+N$). A reader interested in more details is asked to refer to [117, 100, 82].

Formulation (3.232) is a general one, ensuring nominal stability by introducing a penalty component corresponding to the omitted part of the infinite prediction horizon. For $\mathbf{R} = \mathbf{0}$, we obtain a case which corresponds to the form of the cost function most frequently used in MPC controllers. The mentioned penalty component for the terminal state is defined by process state variables, it was calculated using the process model in the form of state equations, see (3.231). Therefore, if we want to apply the presented approach directly to MPC algorithms using input-output type models (DMC, GPC), it is necessary, in order to calculate the penalty component, to apply a process description by state variables or an equivalent one.

Analyzing a DMC algorithm, it is easy to note that, due to the finite length $D$ of step responses, the predicted output values become stabilized after $N_u + D - 1$ sampling instants. It follows from (3.23) and (3.22) that

$$y^0(k+p|k) = y(k) + \sum_{j=1}^{D-1} (\mathbf{S}_D - \mathbf{S}_j) \triangle u(k-j) = y^{0\infty}(k), \quad p \geq D-1$$

$$\triangle y(k+p|k) = \sum_{j=1}^{N_u} \mathbf{S}_D \triangle u(k+N_u-j|k) = \triangle y^{\infty}(k), \quad p \geq N_u + D - 1$$

Therefore, denoting $y^{\infty}(k) = y^{0\infty}(k) + \triangle y^{\infty}(k)$, we have

$$J(\infty) = \sum_{p=N_1}^{N_u+D-2} \left\| [y^{sp}(k+p|k) - y^0(k+p|k)] - \triangle y(k+p|k) \right\|_{\mathbf{\Psi}}^2 + \\ + \lim_{p\to\infty} [p - (N_u + D - 2)] \left\| [y^{sp\infty}(k) - y^{\infty}(k)] \right\|_{\mathbf{\Psi}}^2 + \\ + \sum_{p=0}^{N_u-1} \left\| \triangle u(k+p|k) \right\|_{\mathbf{\Lambda}}^2$$

where we assumed

$$y^{sp}(k+p|k) = const. = y^{sp\infty}(k), \quad p \geq N_u + D - 1$$

It follows from the obtained dependence that the control input which minimizes the cost function $J(\infty)$ must ensure equality $y^{\infty}(k) = y^{sp\infty}(k)$, as long as such feasible control input exists and $\mathbf{\Psi} > 0$.

Therefore, an *equivalent formulation* of the DMC algorithm with an infinite prediction horizon would be:

- to take in the optimization problem the cost function with a finite prediction horizon equal to $N = N_u + D - 2$ and
- to add to the set of constraints the equality constraint $y^{\infty}(k) = y^{sp\infty}(k)$, *i.e.*, the constraint

$$y^0(k+N_u+D-1|k)] + \triangle y(k+N_u+D-1|k) = y^{sp\infty}(k)$$

In situations of shorter dynamics horizons $D$, the proposed formulation can turn out to be not practical, from a point of view of robust calculations. When there are constraints on process control inputs, then adding the above output constraint may cause the feasible set of the optimization problem becomes empty, at certain sampling instants. Special measures should then be undertaken to cope with such events (see next chapter). One such measure is to introduce the critical constraint into the cost function, in the form of a penalty term, *e.g.*, using cost function in the following form

$$J^{\rho}(\infty) = \sum_{p=N_1}^{N_u+D-2} \left\| [y^{sp}(k+p|k) - y^0(k+p|k)] - \triangle y(k+p|k) \right\|^2_{\mathbf{\Psi}} + \\ + \rho \left\| [y^{sp\infty}(k) - y^{\infty}(k)] \right\|^2_{\mathbf{\Psi}} + \sum_{p=0}^{N_u-1} \left\| \triangle u(k+p|k) \right\|^2_{\mathbf{\Lambda}}$$

where penalty coefficient value $\rho$ should be selected in a way to ensure satisfaction of the equality $y^{\infty}(k) = y^{sp\infty}(k)$ with a desired accuracy. Assuming $\mathbf{\Psi}(p) = \mathbf{\Psi}$ for $p = 1, ...N_u + D - 2$, $\mathbf{\Psi}(N_u + D - 1) = \rho\mathbf{\Psi}$ the cost function formulated above can be presented in an equivalent form, which is standard for the DMC algorithm (see Section 3.2)

$$J^{\rho}(\infty) = \sum_{p=N_1}^{N_u+D-1} \left\| [y^{sp}(k+p|k) - y^0(k+p|k)] - \triangle y(k+p|k) \right\|^2_{\mathbf{\Psi}(p)} + \\ + \sum_{p=0}^{N_u-1} \left\| \triangle u(k+p|k) \right\|^2_{\mathbf{\Lambda}} \qquad (3.233)$$

Observe that the proposed methods of calculating the value of the cost function with an infinite horizon in the DMC algorithm leads to larger dimensionality of matrices $\mathbf{M}^P$ and $\mathbf{M}$, see (3.38) and (3.39), in cases of large values of the process dynamics horizon $D$. These matrices are calculated for a finite prediction horizon $N = N_u + D - 1$. Therefore, a selection of possibly large sampling period is important, as this results in a smaller value of $D$.

The GPC algorithm with an infinite prediction horizon, called GPC$^{\infty}$, was analyzed by Scokaert in [125]. He defined the prediction horizon as $N = \max\{N_u + n_B, n_A, n_C\}$, where $n_B$, $n_A$ and $n_C$ are parameters of the model used in the GPC algorithm, see (3.82). And as a state error (corresponding to $x(k+p|k) - x^{ss}$) he proposed to take a vector of control errors $y^{sp} - y(k+p|k)$ in $\bar{n}_A$ subsequent sampling instants, beginning from $(k+N|k)$, where $\bar{n}_A$ is the degree of a stable part of the polynomial $A(z^{-1})$, present in the process model. The adopted way of the state definition led Scokaert to a quite complex formulation of an algorithm on a finite horizon, with an additional equality constraint. The reader interested in details of realization of the GPC$^{\infty}$ algorithm is asked to refer to [125].

It should be emphasized that the design of MPC algorithms which guarantee nominal stability has allowed, first of all, to gain better justification and understanding of mechanisms influencing stability of the resulting feedback control loop. These mechanisms are, as already observed empirically, a lengthening of the prediction horizon and an introduction of constraints on terminal state. Results which concern stability have also affected modifications or augmentations of later versions of commercial products, which have been used successfully for many years.

**Stable MPC Algorithms with Nonlinear Models and Constraints**

As it was discussed in the introduction, presentation of stability issues of predictive control is intentionally limited to basic mechanisms influencing the stability, first of all for applications in the algorithms presented earlier in this book. Thus, this is by no means a full review of all aspects of this subject, which is very broad, especially when nonlinear process models are considered.

We have already presented and explained basic mechanisms which ensure nominal stability: introduction of terminal constraints on the process state enforcing its value at the end of the prediction horizon, and extension of the prediction horizon to infinity. Explanations, concerning both linear and nonlinear processes, were general. We have drawn attention to the fact that enforcing terminal state values may often be too restrictive, especially in the case of shorter horizons. On the other hand, when extending the prediction horizon to infinity, it is necessary to have an effective method of calculating the value of the cost function on such a horizon. The calculation method, based on replacing the cost corresponding to the terminal infinite section of the horizon with a suitable cost of the terminal state, has been given for linear process models, in the form of state equations. Unfortunately, for nonlinear processes the problem is much more difficult. Thus, there is a series of proposals of nonlinear algorithms, of different levels of complexity, using the following basic mechanisms, in different configurations, to obtain nominal stability and numerical feasibility:

- an additional set constraining terminal states – a relaxed version of the condition enforcing a specified terminal state values at the end of the prediction horizon,
- a cost of the terminal state – an equivalent of the value of the cost function calculated from the end of a final prediction horizon to infinity,
- a stabilizing linear state-feedback, designed for a considered equilibrium point of the nonlinear process, usually for a linear approximation of the process at this point.

A survey of all stable predictive algorithms for nonlinear processes with constraints on process inputs and states (outputs) is far beyond the scope of this book. A reader interested in the subject should refer to a brilliant and extensive review article by Mayne *et al.* [94] containing full bibliography

of the topic, see also [98, 1, 82, 123, 115]. In the following, we shall only briefly comment on a few selected papers, which are, in the author's opinion, representative for the development of the subject and are formulating new concepts.

Undoubtedly, papers [66, 24, 63] are pioneering. It was shown there that enforcing the value of the terminal state guarantees stability (in the last of them, concerning discrete-time control, additional constraints on both inputs and state were considered). An interesting analysis of stability and basic features of predictive control systems is also contained in [104].

In [97] Michalska and Mayne proposed an algorithm where, instead of enforcing the terminal state, a set of admissible terminal states $W_\alpha$ was introduced, being a certain neighborhood of the process equilibrium point. The concept of the algorithm is as follows: first, using a predictive algorithm with a finite horizon bring the state of the process, in a finite number of steps, to the set $W_\alpha$; then, within this set, the control is switched to linear state-feedback, bringing the process to the desired equilibrium state – thus the name given by the authors: "*dual mode algorithm*". In the algorithm discussed, the length of the finite prediction horizon is variable, constantly updated.

Chen and Allgöwer [25] proposed an algorithm with a *quasi-infinite horizon* (for continuous time, similarly as in [97]), where all three above mentioned mechanisms of ensuring stability are used. The control input signal is calculated on-line, by solving an optimization problem on a finite horizon, containing both terminal state constraints and a quadratic penalty term for the terminal state in the cost function. This term is calculated as an upper-bound of the cost function value on the horizon from the terminal state to infinity, with the control input signal generated by a linear state-feedback. As a result of such off-line calculations, the matrix of the quadratic penalty term is evaluated. An interesting feature of the algorithm is that it provides stability, even if a global minimum is not found by the optimization. However, it is necessary to find an admissible solution at every sampling instant – a feature called "*feasibility implies stability*". This feature has been originally proven in [127], for suboptimal versions of the classical discrete-time algorithm with a terminal state constraint and for the dual-mode algorithm [97], at every sampling instant assuming feasibility of solutions and requiring only a certain improvement of the cost function.

A discrete-time algorithm of a structure almost analogous to the one considered in [25] was analysed in a series of papers by De Nicolo, Magni and Scattolini, see, *e.g.*, [102]. The value of the penalty term for the terminal state is here, however, obtained by a direct calculation of a sum of the cost function values starting from the terminal time $N$ and up to some arbitrary (sufficiently remote) time $M$, with control input signal evaluated over this time interval by a linear state-feedback. The disadvantages of this approach are a larger calculation effort and arbitrariness in the selection of the value of $M$. In [83] the algorithm features were improved by giving a recipe for the value of $M$ and lowering the calculation effort by introduction of the control horizon $N_u \leq N$,

analogously as in typical MPC algorithms with linear models. Over the time interval from the current sampling instant $k$ to $k + N_u - 1$ the control input signal is evaluated as a solution of an associated optimization problem, and over the time interval from $k + N_u$ to $k + N - 1$ the control input is given by a linear state-feedback, designed off-line.

A quite effective algorithm, in terms of calculations which are necessary to be performed on-line, is described in [19], for a problem with constraints imposed on control input values only. The idea is to use an optimal linear state-feedback on the whole prediction horizon and to perform on-line optimization in order to calculate corrections to the obtained control inputs only, needed to ensure feasibility. This is in fact an application of the pre-stabilization concept, see [124, 123].

The concise review, presented above, shows that predictive control algorithms based on constrained optimization of nonlinear process models, create a very interesting field of scientific activity. In spite of the fact that basic mechanisms of ensuring nominal stability have already been identified, this algorithms have not yet gone beyond the stadium of theoretical investigations, illustrated by examples of computer simulations. They have not yet reached the stage that would allow to convince potential users to serious, industrial applications.

In this situation, it is most reasonable to put into practice algorithms, which are direct extensions of the industrially proven algorithms with linear models, such as the algorithms with on-line linearizations of nonlinear models, discussed in Section 3.5. They are intuitively convincing and well verified by many simulation experiments. Theoretical analysis of these algorithms is not easy, because they are suboptimal. However, after extending basic structures of these algorithms by adding terminal constraint sets and additional improvement tests, and using the ideas from [127], it occurred possible to obtain nominal stability results: in [93] for an extended nonlinear fuzzy DMC algorithm, and in [74] for a dual-mode type algorithm with a nonlinear process model. In both algorithms only quadratic programming problem, stemming from current linearization of the nonlinear model, is solved at each sampling instant.

### 3.6.2 Feasibility of Constraint Sets, Parameter Tuning

#### Feasibility of Constraint Sets

In Section 3.1 different classes of constraints which may be present in predictive control algorithms were discussed: constraints on values and on increments of process control inputs, (3.7) and (3.8), constraints on controlled outputs (3.9) and on constraint outputs (3.10). In cases when both input and output constraints are present, it may happen that the set of solutions of the controller optimization problem becomes empty, at certain sampling instants. For example, after a strong change in a disturbance value it may not be possible,

while keeping constraints on the control inputs preserved, to bring the process output to a value satisfying constraints earlier than after several sampling instants, *i.e.*, it may be impossible to satisfy the output constraint (3.9) or (3.10) at sampling instant $k + N_1$ and at some subsequent ones. A possible way to deal with this issue is to use the concept of the so-called *constraint window*.

Using the constraint window consists in considering, when calculating the control input at sampling instant $k$, the output constraints only over a subinterval $[N_{cw1}\ N_{cw}]$ of the prediction horizon, where values $N_{cw1}$ and $N_{cw}$, $N_1 < N_{cw1} < N_{cw} \leq N$, define the width of the constraint window. An appropriately large value $N_{cw1}$ should ensure satisfaction of all constraints at each sampling instant over the constraint window, and the value $N_{cw}$ should be such such that to ensure satisfaction of all the constraints to the end of the prediction horizon (even outside the window). For a stable linear process Rawlings and Muske proved in [117] existence of such vales $N_{cw1}$ and $N_{cw}$, that for control input constraints $\mathbf{G}u(k+p|k) \leq g$, $p = 0, 1, ...$, the output constraints $\mathbf{H}x(k+p|k) \leq h$ (represented equivalently by state variables) are satisfied for $p = N_{cw1}, ..., N_{cw}$ and for $p > N_{cw}$, where $\mathbf{G}$ and $\mathbf{H}$ are matrices defining the constraint functions. Of course, a constraint window type strategy does not remove constraint violations at first instants of the prediction horizon and can be applied, if such uncontrolled violations over short time periods are unavoidable and can be tolerated in the process.

A frequently used practical approach to solve the problem of ensuring that the constraint set of the controller optimization problem becomes non-empty is to treat all or some of the constraints on output values as *soft constraints*. Such constraints can be violated, but this violation is appropriately penalized by penalty components added to the cost function, in such a way that it actually occurs only in situations when the constraint set becomes empty.

Recall the predictive controller optimization problem, formulated in the form (3.70), standard for numerical methods – but with matrix $\mathbf{A}$, which defines the inequality constraints, divided into two parts: the one corresponding to constraints on control inputs and the other corresponding to constraints on outputs, respectively:

$$\begin{aligned} &\min\{J(x) = \frac{1}{2}x^T\mathbf{H}x + f^T x\} \\ &\text{subj. to:}\quad x_{\min} \leq x \leq x_{\max} \\ &\qquad\qquad \mathbf{A}_u x \leq b_u \\ &\qquad\qquad \mathbf{A}_y x \leq b_y \end{aligned} \tag{3.234}$$

where $\mathbf{A}^T = [\mathbf{A}_u^T\ \mathbf{A}_y^T]^T = [-\mathbf{J}^T\ \mathbf{J}^T\ -\mathbf{M}^T\ \mathbf{M}^T]^T$, $b^T = [b_u^T\ b_y^T]^T$, compare with (3.70). Relaxing the constraints $\mathbf{A}_y x \leq b_y$ can be obtained by formulating the optimization problem (3.234) in the following form

$$\begin{aligned} \min_{x,v}\{J(x) = \frac{1}{2}x^T\mathbf{H}x + f^T x + \rho_i \sum_{i=1}^{n_v} v_i^2\} \\ \text{subj. to: } x_{\min} \le x \le x_{\max} \\ \mathbf{A}_u x \le b_u \\ \mathbf{A}_y x \le b_y + v \\ v \ge 0 \end{aligned} \tag{3.235}$$

where $v = [v_1 \ \cdots \ v_{n_v}]^T$ is a vector of additional variables of dimension $n_v$ equal to the number of relaxed constraints, $n_v = \dim b_y$, while $\rho_i > 0$ are penalty coefficients, which can be generally different for different constraints. Penalty terms need not to be quadratic, instead of $\sum_{i=1}^{n_v} v_i^2$ it is possible to use $\sum_{i=1}^{n_v} v_i$ – which is a formulation of the *exact penalty function*. This function becomes exact, *i.e.*, constraints are not exceeded for finite, appropriately large values of penalty coefficients $\rho_i$, in cases when the constraint set is non-empty. But the choice of appropriate values of the penalty coefficients is more difficult.

Since, in practice, the importance of satisfying individual constraints on outputs can differ, then it is also possible to use appropriate sequencing of the constraints in the order of their importance, *i.e.*, a *prioritization* of the constraints. An example of such a simple prioritization would be an introduction of different penalty coefficients for different constraints in the formulation of a relaxed optimization problem (3.235). In certain problems it may be reasonable to treat only constraints of lower priorities as soft constraints, or sequencing within soft constraints properly classified. This can be organized by subsequent checking if the constraint set is non-empty, starting treating all the constraints as hard, and then relaxing some of them beginning with those with lowest priorities, until the constraint set of the resulting optimization problem becomes feasible (non-empty) [146].

**Parameter Tuning**

The question of importance and selection of appropriate values for basic parameters of the predictive control algorithms will now be discussed. This should be quite satisfactory for algorithms with linear models, like the DMC, GPC and MPCS algorithms discussed in previous sections of this chapter (for the MPCS algorithm, coefficients of the state observer matrix are also tunable parameters, see Section 3.4). In cases of algorithms with nonlinear models there are usually additional parameters connected with the type of a considered algorithm, such as, *e.g.*, number of fuzzy sets and their parameters, present in rule antecedents of TS fuzzy predictive controllers, as discussed in Section 3.5.

Basic parameters of a general MPC algorithm, generating control input values as a result of an optimization of the cost function (3.1), are:

- $N$ — prediction horizon,
- $N_u$ — control horizon, $N_u \le N$,
- $\mathbf{\Psi}(p), \mathbf{\Lambda}(p)$ — matrices of weighting coefficients in the cost function,

- $N_{cw1}, N_{cw}$ — limits of the constraint window (or other parameters defining a strategy ensuring that the feasible set is non-empty at every sampling instant, *e.g.*, penalty coefficient values $\rho_i$ in (3.235)),
- $T_p$ — sampling period (time between two subsequent controller interventions),
- $\gamma$ — coefficient of a linear filter defining the shape of the reference trajectory.

It is also generally possible to treat the value $N_1$, $N_1 \geq 1$, as an additional algorithm parameter. It defines a beginning of a (sub)interval of the prediction horizon over which the predicted control error is taken into account, in the first sum in the cost function. Usually $N_1 = \tau + 1$ is assumed, where $\tau$ denotes a delay in the process step response, given as a number of sampling periods (number of first coefficients of discrete step response equal to zero). Selection of $N_1 < \tau + 1$ does not lead to errors – but does not make much sense, as predictions of process outputs for first $\tau$ instants (*i.e.*, for $k+p \leq k+\tau$) do not depend on decisions (actual and future control inputs) evaluated at sampling instant $k$. Values $N_1 > \tau + 1$ are also possible, this is related to the concept of the constraint window discussed in the previous point.

The value $N$ describes the length of the prediction horizon. The infinite horizon ensures nominal stability for all values of the remaining parameters, provided the solution set of the optimization problem is always non-empty, as it was presented in Section 3.6.1. If the prediction horizon $N$ is long enough, when compared to the process dynamics and the feedback control system operates in a correct way, then further enlargement of $N$ usually does not lead to improvements – properties of the control system become insensitive to the value of $N$.

Sensitivity of a control system to the length of the control horizon $N_u$ is relatively low, as long as this horizon is not too short [98, 123]. Since the value $N_u$ decides on the number of decision variables in the optimization problem (equal to $N_u \cdot n_u$), then it is a good practice to take relatively small values of $N_u$. This is especially significant in case of NO type algorithms (with on-line nonlinear optimization). Extremely small values are possible, up to $N_u = 1$, as it is demonstrated in an example below (a discussion of such cases can be found *e.g.*, in [82, 123]).

**Example 3.13**
Consider again the example of a strongly nonlinear reactor with a polymerization process, presented in Example 3.8. It will be demonstrated now how operation of the control system with the MPC-NO algorithm changes when the length of the prediction horizon $N$ and the length of the control horizon $N_u$ are varied.

Figures 3.69 and 3.70 present trajectories of process output and input variables for the control horizon $N_u = 2$ and prediction horizons, subsequently, $N = 2$, 3, 5 and 10. A small difference between trajectories for $N = 5$ and for $N = 10$ attracts attention. The result for $N = 10$ was given as an example

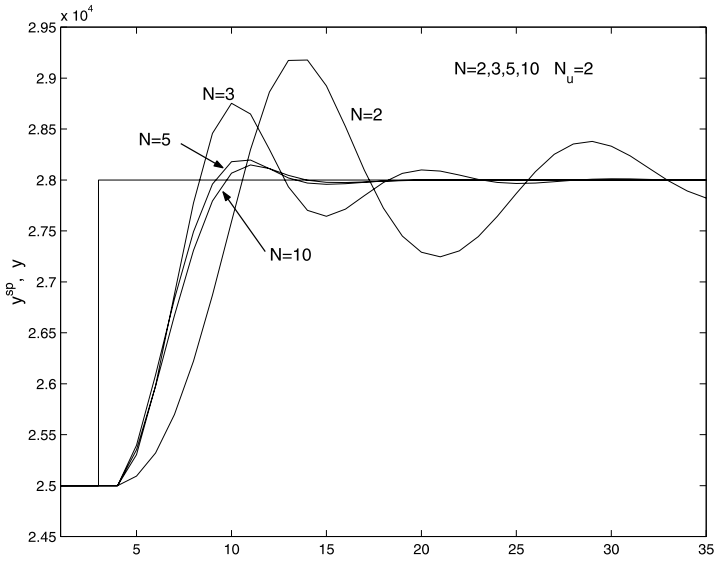


**Fig. 3.69.** Trajectories of the controlled output $y$ for different values of the prediction horizon $N$, for the control horizon $N_u = 2$

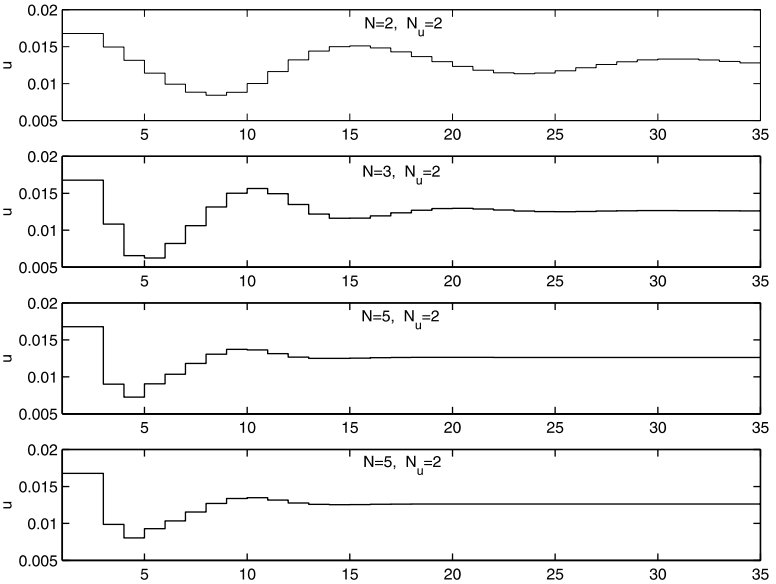


**Fig. 3.70.** Trajectories of the control input $u$ for different values of the prediction horizon $N$, for the control horizon $N_u = 2$

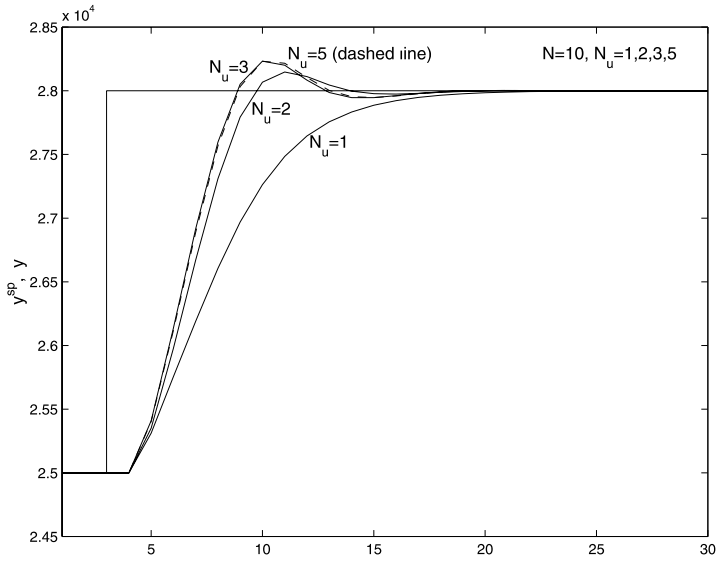


**Fig. 3.71.** Trajectories of the controlled output $y$ for different values of the control horizon $N_u$, for the prediction horizon $N = 10$

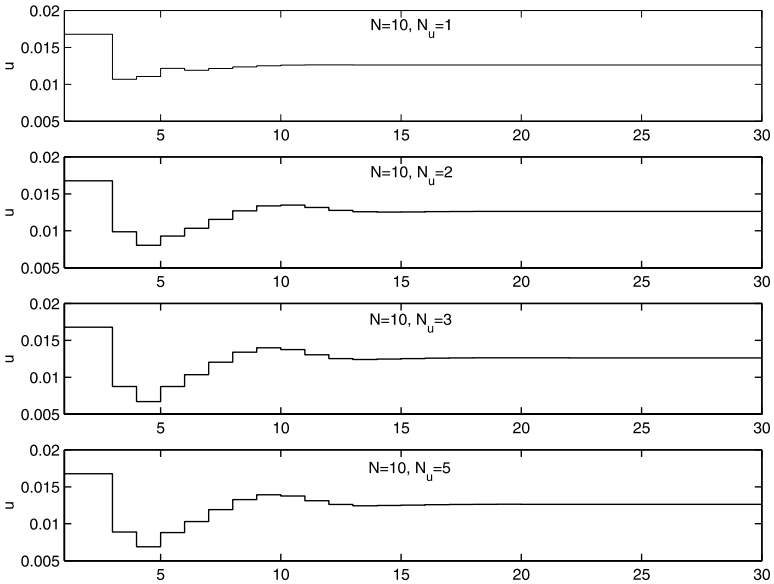


**Fig. 3.72.** Trajectories of the control input $u$ for different values of the control horizon $N_u$, for the prediction horizon $N = 10$

of the prediction horizon large enough to make the control system insensitive to increase of $N$. Behavior of the system with larger values of $N$ was also examined, but the obtained trajectories were practically the same as those for $N = 10$.

The presented results of simulation experiments clearly show a stabilizing effect of longer prediction horizons. They also explain why in Example 3.8 the value $N = 10$ was assumed, while much shorter control horizon $N_u = 2$ was taken.

Figures 3.71 and 3.72 present trajectories of process output and control input variables for the prediction horizon $N = 10$ and for control horizons $N_u = 1, 2, 3, 5$. There is clearly a qualitative difference between trajectories obtained for $N_u = 1$ and for higher values. There is only a quantitative difference between trajectories obtained for $N_u = 2$ and $N_u = 3$, and a not very large one. There are practically no differences between trajectories for $N_u = 3$ and for higher values – trajectories for $N_u = 3$ and, as an example of higher values, for $N_u = 5$ differ only slightly. The control system for several larger values of $N_u$ was also simulated, obtaining results which are undistinguishable from those obtained for $N_u = 5$.

The presented results are rather typical: they indicate that it is enough, generally, to take relatively small values of $N_u$, as trajectories in predictive control systems quickly become insensitive to an increase in the length of the control horizon. □

The basic parameters affecting dynamics of a predictive feedback control system, and particularly its stability and robustness for a finite length of the prediction horizon, are values of elements of the weighting matrices $\mathbf{\Psi}(p)$ and $\mathbf{\Lambda}(p)$. These matrices are usually assumed in a diagonal form and in most cases they do not depend on $p$, $\mathbf{\Psi}(p) = \mathbf{\Psi}$ and $\mathbf{\Lambda}(p) = \mathbf{\Lambda}$. Relations between values $\psi_{ii}$ of entries of the matrix $\mathbf{\Psi}$ define relative weights of control errors corresponding to individual controlled outputs. On the other hand, entries $\lambda_{ii}$ of the matrix $\mathbf{\Lambda}$ define relative weights of individual control inputs. The most important, however, are relations between elements of matrices $\mathbf{\Psi}$ and $\mathbf{\Lambda}$, as they define relative cost of control errors as compared to the cost of control input moves. Increasing values of $\lambda_{ii}$, while keeping values $\psi_{ii}$ constant, causes damping of control input signals, *i.e.*, leads to a slower but more fluent operation of control systems. Thus, it diminishes the risk of possible activity of inequality constraints, increases stability margins and robustness of control loops. In the simplest and often encountered case when $\mathbf{\Psi}(p) = \mathbf{I}$ and $\mathbf{\Lambda}(p) = \lambda\mathbf{I}$, corresponding to the cost function (3.5), there remains $\lambda$ as a single scalar tuneable parameter. For the operation of algorithms with linear models (DMC, GPC or MPCS) it is the basic tuneable parameter, see, *e.g.*, [18, 123]. The influence of values of $\lambda$ on step responses in a DMC control system was analyzed in Example 3.1 in Section 3.2.2.

Selection of an appropriate value of the sampling period $T_p$ is important and closely connected with the length of horizons. It is most evident in the

case of the DMC algorithm. Selecting too small value for $T_p$ results in a large number of elements of the (finite) step response, *i.e.*, in a long horizon $D$ of process dynamics. This, in turn, leads to larger matrices $\mathbf{M}^P$ and $\mathbf{M}$, see Section 3.2. Thus, we get larger dimensionality, and thus memory requirements and calculation time, of the optimization task, if we want to maintain a similar ratio of prediction and control horizons to the process dynamics horizon. On the other hand, taking too large value of $T_p$ will result in introduction of a too large sampling delay and too rare changes in the control signal, thus leading to poorer behavior of the feedback control loop.

In each MPC algorithm, it is possible to use in the cost function a *reference trajectory* $y^{ref}(k+p|k)$, $p = N_1, ..., N$, in place of a set-point trajectory, $y^{sp}(k+p|k)$, $p = N_1, ..., N$. The reference trajectory was already used in the *Model Predictive Heuristic Control* (MPHC) algorithm [119, 120], known later under the name of MAC (*Model Algorithmic Control*). A first-order reference trajectory can be generally defined as follows

$$y^{ref}(k+p|k) = \gamma y^{ref}(k+p-1|k) + (1-\gamma)y^{sp}(k+p|k), \quad p = 1, ..., N$$

where $y^{ref}(k|k) = y(k)$, $0 \leq \gamma < 1$, see *e.g.*, [17, 18].

For the most important case when a constant set-point value is assumed over the whole prediction horizon, $y^{sp}(k+p|k) = y^{sp}(k)$, $p = 1, ..., N$, at every sampling instant the entire reference trajectory is uniquely defined by the current values $y(k)$ and $y^{sp}(k)$, and thus it can be presented in a slightly more convenient form

$$\begin{aligned} y^{ref}(k+p|k) &= (1-\gamma^p)y^{sp}(k) + \gamma^p y(k) \\ &= y^{sp}(k) - \gamma^p[y^{sp}(k) - y(k)], \quad p = 1, ..., N \end{aligned} \tag{3.236}$$

where $y^{sp}(k) - y(k)$ is the control error. It can be easily seen from (3.236) that the control error is required to be reduced to zero exponentially over the prediction horizon, with the exponent defined by $\gamma$. An example of such a reference trajectory is presented in Fig. 3.73.

When applying the reference trajectory, the cost function of the MPC algorithm takes the form

$$J(k) = \sum_{p=N_1}^{N} \left\| y^{ref}(k+p|k) - y(k+p|k) \right\|^2_{\mathbf{\Psi}(p)} + \sum_{p=0}^{N_u-1} \left\| \triangle u(k+p|k) \right\|^2_{\mathbf{\Lambda}(p)} \tag{3.237}$$

where values $y(k+p|k)$ constitute the trajectory of predicted outputs and $\triangle u(k+p|k)$ are decision variables, compare with (3.1). Application of the reference trajectory transforms, initially, the trajectory of set-points to a continuous, smoother form changing as the exponential pattern, defined by $\gamma$. The value of $\gamma$, $0 \leq \gamma < 1$, is a tuneable parameter of the algorithm, for $\gamma = 0$ the reference trajectory becomes the set-point trajectory $y^{sp}(k+p|k)$, $p = 1, ..., N$. Along with the increase in $\gamma$, convergence of the reference trajectory to the required set-point value becomes slower, smoother – thus requirements for control signal changes become relaxed. A similar effect could

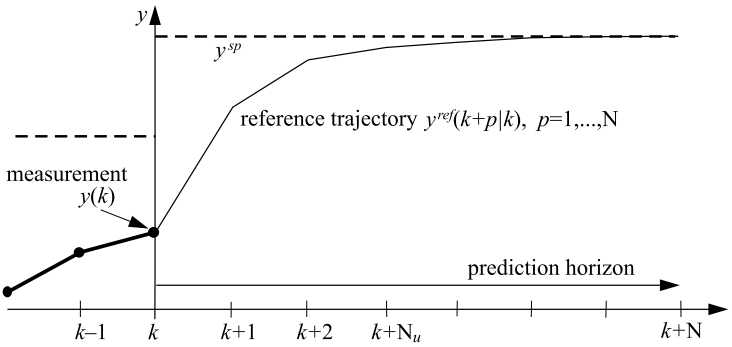


**Fig. 3.73.** Illustration of the concept of a reference trajectory, for the set-point value constant over the prediction horizon

be obtained without a reference trajectory, only with a right selection of elements of the weighting matrices $\mathbf{\Psi}(p)$ and $\mathbf{\Lambda}(p)$. Especially, as we are tuning relations between control errors and control input increments over the prediction and control horizons, it would be enough to assume $\mathbf{\Psi}(p) = \psi(p)\mathbf{I}$, $\mathbf{\Lambda}(p) = \lambda(p)\mathbf{I}$, and to select appropriate values of $\lambda(p)$ versus $\psi(p)$. Using the reference trajectory may be slightly simpler, we only need one parameter $\gamma$ for tuning.

In case of a step change in the set-point value, using a reference trajectory leads initially to effects similar to those obtained when filtering the set-point values by a first order filter, in a classical feedback control loop with two degrees of freedom, as shown in Fig. 3.74. However, these are not equivalent structures: filtering the set-point value in the system shown in Fig. 3.74 means operation in an open-loop system, while using the reference trajectory in a MPC control system results in the introduction of an additional feedback loop. This is presented in Fig. 3.75, which is valid for any MPC controller (*e.g.*, DMC, GPC, MPCS, nonlinear controllers) using the reference trajectory in the cost function, instead of the trajectory of set-points. The difference between initial set-point filtering and the reference trajectory can be especially clearly seen in the case of a disturbance change, assumed to act on the


**Fig. 3.74.** Classical feedback control loop with two degrees of freedom

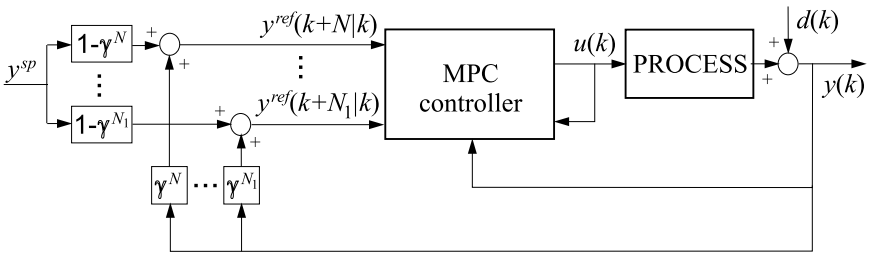


**Fig. 3.75.** Structure of MPC feedback control loop with reference trajectory applied (assuming the set-point constant over the prediction horizon)

process output. In this case, the set-point filter has no influence on trajectories resulting in the control system shown in Fig. 3.74, but the use of the reference trajectory influences the trajectories resulting in the control system from Fig. 3.75

The reference trajectory, treated as one of the practical ways of shaping the behavior of MPC algorithms, is currently rather rarely met in the literature. It is of smaller significance for algorithms with constraints, implemented in a numerical version, with on-line optimization. There are usually no rational reasons to slow down reduction of control errors in the situation of full control over constraints on amplitudes and rates of change of process control inputs. On the other hand, the behavior of the control system itself, its stability and robustness, are affected mainly by a right selection of horizons and weighting coefficients, first of all elements of the matrix $\mathbf{\Lambda}$ (see Example 3.2).

# 4

# Set-point Optimization

## 4.1 Steady-state Optimization in Multilayer Process Control Structure

The task of *set-point optimization* in a multilayer control structure is to provide the directly subordinate feedback controllers of the lower layers with *best possible set-point values*, see Fig. 1.2 in Chapter 1 and the discussion therein. "The best possible set-point values" means optimal dynamic trajectories or optimal constant (steady-state) values of the set-points leading to maximal achievable values of the prescribed economic criteria of the process operation, while keeping the process variables within safe operation limits and satisfying certain additional constraints of a technological nature. The case of steady-state (static) optimization will be considered in this chapter, as it dominates in practice in process industries, for a number of well-known reasons, see *e.g.*, [16].

The controllers under supervision of the optimization unit are the advanced controllers of the constraint control layer and certain controllers of the direct control layer. The location of the subordinate feedback controllers is not essential for the task of the optimization layer, as from this layer the plant is seen together with all feedback controllers, the *feedback controlled plant* is being optimized. However, the information sent from the optimization layer to the constraint controllers is different than that sent to the direct controllers only, from the point of view of the kind of process variables.

Therefore, consider first the case without the constraint control layer, with direct control and optimization layers only, as depicted in Fig. 4.1 (compare with Fig. 1.2).

The steady-state optimization problem can now be formulated as (1.21) or in an equivalent, but formally simpler form (1.22), which is repeated here for convenience

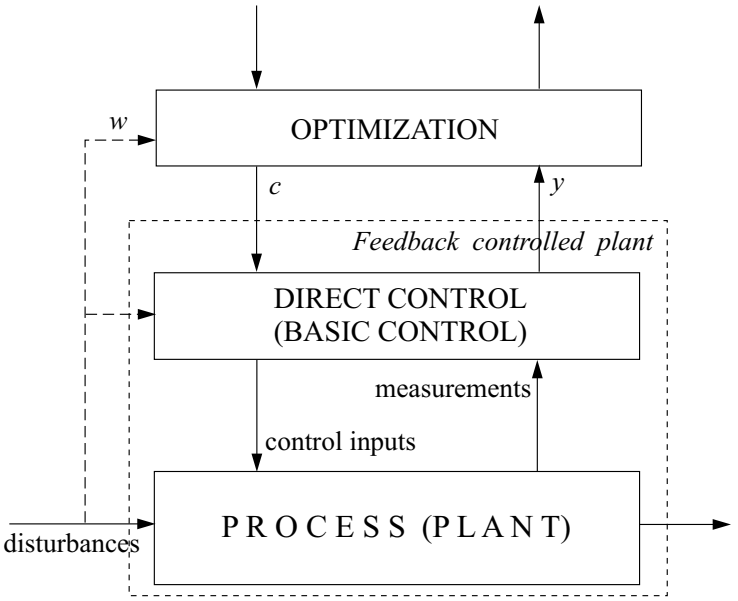


**Fig. 4.1.** Feedback controlled plant as an object for set-point optimization in the two-layer structure (without the constraint control layer)

$$\begin{aligned} &\min Q(c,y) \\ &\text{subj. to:}\ y = F(c,w) \\ &\qquad\quad c \in C \\ &\qquad\quad y \in Y \end{aligned} \tag{4.1}$$

In the above description, $F$ describes the steady-state process model, $c$ denotes the vector of decision variables of the optimization problem, being in the considered case the set-points for the controlled variables – to be optimized, $y$ denotes the vector of process outputs significant for the process economics and constraint satisfaction and $C$, $Y$ are constraint sets, see Section 1.5. Usually $Q(c,y)$ represents instantaneous net production profit. Assuming linear dependence of costs of individual materials on their prices and control (stabilization) of energy streams and certain raw material streams, the performance function (to be minimized) can be formulated in the following simple linear form

$$Q(c,y) = \sum_{j=1}^{n^J} p_j c_j - \sum_{j=1}^{m^J} q_j y_j \tag{4.2}$$

where $p_j$ denote prices of the mentioned input streams, whereas $q_j$ prices of output products and $n^J$ is the number of controlled input streams. The

output variables $y$ may represent production rates of both products and waste materials. In the latter case the prices could have negative sign. There may also exist components of the output vector $y$ not entering the performance function, but important for formulation of the process constraints, like product concentration in the output stream. These process output variables are called *constraint variables*, see Section 3.1 and *e.g.*, [142]. Therefore, the situation when $m^J < \dim y$ is possible and encountered.

In complex processing plants, the feedback (regulatory) control consists now of a direct (basic) dynamic control layer operating usually with PID controllers and a hierarchically higher dynamic constraint control layer (set-point control layer) – called also advanced control layer or MPC control layer, as model predictive control (MPC) algorithms are usually applied there, see Chapter 1 and [62, 134, 11, 115, 16]. Set-points for the constraint controllers are the desired steady-state values of constrained, feedback controlled process outputs, see Fig. 1.3 in Chapter 1, where the vector of these process outputs was denoted by $y^d$ (being, in general, a sub-vector of the overall output vector $y$), and where the vector of decision variables of the optimization problem was divided into two parts, $c = (c^f, c^d)$. The first sub-vector represents set-points for direct controllers, while the second optimal steady-state values of variables corresponding to the outputs of constraint controllers – both calculated on the basis of the steady-state process model used at the optimization unit, including the available information about current values of disturbances $w$. The steady-state optimization problem (4.1) can in this case be rewritten in a more detailed form, as follows

$$\begin{aligned} &\min Q(c^f, c^d, y^f, y^d) \\ &\text{subj. to:}\ (y^f, y^d) = F(c^f, c^d, w) \\ &\qquad c^f \in C^f,\ c^d \in C^d \\ &\qquad y^f \in Y^f,\ y^d \in Y^d \end{aligned} \tag{4.3}$$

After every solution of the optimization problem, the optimal value of the feedback constrained outputs $y^d$ is transmitted to the constraint controller, as its set-point. Transmission of optimal values of $c^d$ is not necessary if $\dim c^d = \dim y^d$, as there is then no freedom in the constraint controller operation, although this information may be useful when changing the operating points. Therefore, it was marked by a dashed arrow in Fig. 4.2, where the described structure is presented (compare with Fig. 1.3 and Fig. 4.1). However, if the MPC constraint controller has more degrees of freedom, *i.e.*, $\dim c^d > \dim y^d$, then this knowledge will be necessary for its economically optimal operation.

In the static optimization problem (4.1) (and (4.3)) there occurs the vector $w$ representing uncontrolled process input values (disturbances), in particular those which are significant for the process optimality. In reality, not all disturbances can be measured. Moreover, measurements even if available, are corrupted by errors. Therefore, disturbances must be appropriately modeled and only certain estimates of their values are available. The kind of these

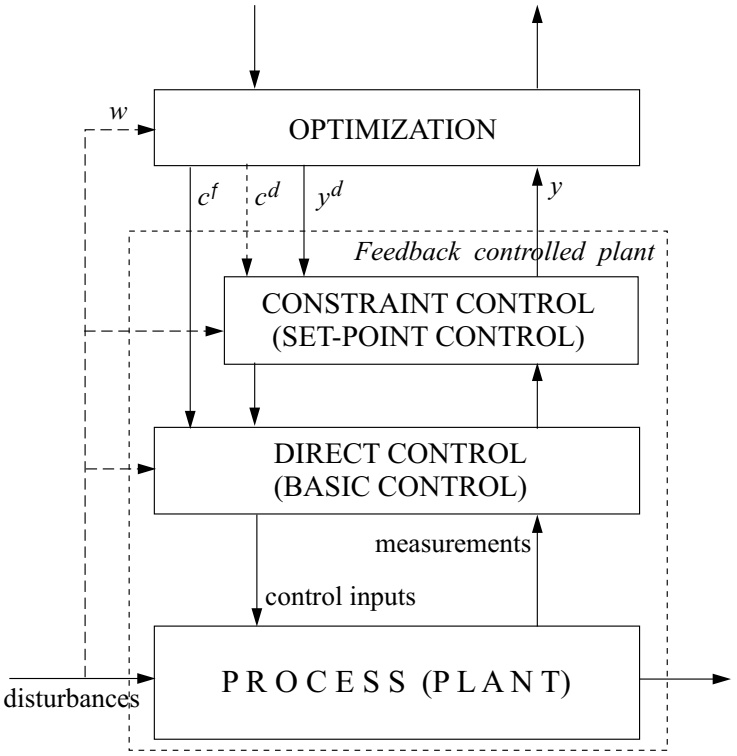


**Fig. 4.2.** Feedback controlled plant as an object for set-point optimization in the multilayer structure with both direct control and constraint control layers

estimates can be different, depending on disturbance nature and available knowledge. In particular, most widely used disturbance models are deterministic, stochastic or set-bounded. We will not present this topic here, as deterministic description is dominating in process control models for on-line steady-state optimization purposes. However, the interested reader is referred to [16], where this subject is covered in more detail.

In a classical multilayer structure it is usually assumed that the disturbances considered at the optimization layer are slow-varying, when compared to the controlled process dynamics. Then the steady-state optimization problem can be solved with a frequency much lower than the sampling frequency of directly subordinate feedback controllers, and the time interval between two successive optimizations is much longer than the time of a step-response transient processes in the controlled plant.

However, if variability of the disturbances significant for the process optimality is not so slow, in the worst case even comparable with the process

dynamics, then the values of the set-points should be updated more often. It is especially important in cases when predictive controllers are applied, as then efficient solutions to this problem are possible. This topic will be covered in the next section.

From a point of view of the optimization, the disturbances $w$ are parameters. But the problem is that the model of the process $F$ should describe the reality well. Therefore, values of these disturbances should be known – measured or estimated. Only then the solution of the steady-state optimization problem with the process model does determine the operating point (set-point) which is optimal for the real process. More precisely, it is then close enough to the real optimal, but not strictly optimal due to unavoidable uncertainties and inaccuracies. If the case is not so, then the point evaluated using an (only rough) process model can be far from the optimal one for the real process. In situations with significant uncertainty, but when the disturbances can be treated as constant during longer time intervals (significantly longer than the controlled process settling time), the approach based on an iterative use of additional measurement information from the plant allows to improve optimality of the set-points, see *e.g.*, [16]. This will also be discussed in this chapter.

## 4.2 Steady-state Optimization for Model Predictive Control

In complex processing plants the task of optimization usually splits into more than one layer – it consists of a higher global *plant-wide optimization* and several lower *local steady-state optimizations* (LSSOs) connected with individual technological units of the plant. Each layer operates at a different frequency, a typical sampling interval of the basic control is about a second, of the MPC control (constraint control) about a minute, while LSSO may be repeated every hour and global plant-wide optimization every day, see *e.g.*, [115].

The local steady-state optimizer uses a comprehensive nonlinear steady-state model of the underlying process, performing constrained optimization of an economic objective function. It adjusts set-point values taking into account changes in slowly (or rarely but abruptly) varying disturbances and in requirements of the higher layer (plant-wide optimization). Such an approach is reasonable and leads to process operation close to optimal if variability of disturbances that influence economic effectiveness of the process is much slower than the dynamics of the feedback controlled process itself. However, for many processes this condition is not fulfilled at all or during significant part of the time, due to variability of process disturbances, like flow rates and properties of feed and energetic streams. Operation in the classical hierarchical structure can then result in a significant loss of economic effectiveness.

On the other hand, application of MPC algorithms is becoming more and more popular, they dominate at the constraint control layer, which is therefore

even called the MPC layer [62]. In general, an MPC algorithm computes at every sampling instant, using a dynamic process model, a sequence of optimal adjustments of its future output variables, in order to minimize predicted control errors over certain prediction horizon. Then only the first element of the sequence is actually applied, as the whole procedure is repeated at the next sampling instant, see Chapter 3.

To evaluate predicted control errors, the MPC algorithm must be supplied with desired set-points, which can be called *target set-points* [62]. When the disturbances vary slowly or rarely and thus the process can operate at unchanged optimal steady-states over longer time periods, this target set-points come directly from LSSO. However, this is not the case for many processes, as stated earlier. Then keeping the set-point constant over many sampling instants, waiting for the next calculation of LSSO despite changes in disturbances, would lead to economic losses. Certainly, the most obvious and successful approach would be to perform the LSSO as often as needed, even at each sampling instant of the MPC controllers. However, recall that the LSSO uses a comprehensive nonlinear steady-state model of the process, thus it performs a nonlinear constrained optimization. This may be a difficult and time-consuming task, not possible to be executed on-line at each or even at every few sampling instants. Therefore, a simpler approach became an industrial practice: the use of an additional *steady-state target optimization* coupled with the MPC algorithm, see *e.g.*, [62, 11, 115]. The resulting control structure is depicted in Fig. 4.3, where direct influence of the optimization unit on the direct control layer is omitted, as it does not lead to loss of generality and is a common case. As usual in a multilayer structure, each control unit (functional block) calculates its output with a different frequency, the higher the block in the structure the lower the frequency (MPC steady-state target optimization and MPC dynamic optimization constitute one control unit).

The goal of *MPC steady-state target optimization* (SSTO) is to recalculate the steady-states available from the local steady-state optimization (LSSO) every time the MPC controller executes. Since the SSTO runs with the frequency of the MPC, it must use a simple process model, in practice it is usually a linear model. In most cases a steady-state version of the linear dynamic model (*i.e.*, the gains matrix) used in the MPC algorithm is recommended and reported to be used [62, 11, 115]. This solution is simple and results in a linear programming (LP) problem, if the economic objective function is linear, as it is in most cases. The LP problem can be solved at each sampling instant, but the approach may lead to losses of economic optimality, since the mentioned linear model may be significantly different from the comprehensive one used in the LSSO, for most of the operating points. The problem has been recognized, *e.g.*, in [62] it is proposed to estimate and treat explicitly uncertainty in the steady-state gain matrix, in the framework of a robust target steady-state calculation.

On the other hand, Qin and Badgwell [115] report solutions using linearizations of the comprehensive nonlinear model instead of the gain matrix of

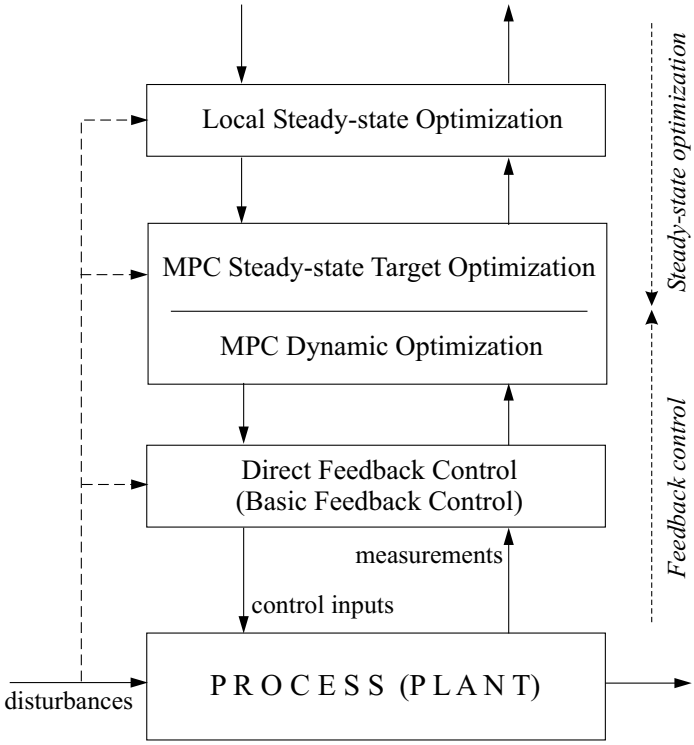


**Fig. 4.3.** Control structure with MPC constraint control layer with steady-state target optimization (recalculation)

the dynamic one and applications of quadratic programming (QP) instead of LP. Further, in [138] process models resulting from a quadratic and from a piecewise-linear approximations of the comprehensive nonlinear model at each sampling instant are proposed to be used in target steady-state optimizations (the latter occurred to be the only successful in a reported example problem). The described approaches are based on the reasonable argument that the steady-state process models used for the MPC SSTO should be consistent with the comprehensive nonlinear model used in the LSSO, rather than with the linear dynamic model from the MPC algorithm. This is also combined with the recognition that increase in computational power of microprocessors makes it possible to use more computationally demanding algorithms for MPC SSTO. The best solution would certainly be, as mentioned before, to repeat the nonlinear local steady-state optimization every time the MPC controller executes, thus eliminating the need for steady-state target recalculation. This

solution may now be possible, but still only in rather limited cases, for very slow processes with relatively simple nonlinear steady-state models.

### 4.2.1 MPC Steady-state Target Optimization

Consider first a most standard situation when numbers of controlled variables $y$ and manipulated variables $u$ of the MPC controller are equal, *i.e.*, $\dim c^d = \dim y^d = \dim y^{sp}$, where $y^{sp}$ denotes set-points for $y$ in the MPC controller formulation (see Chapter 3). Certainly, $\dim u = \dim c^d$, as they correspond to the same physical variables. Denoting, as in Chapter 3, the prediction horizon by $N$, the control horizon by $N_u$, by $y(k+p|k)$ prediction of the output $y$ for a future sampling instant $k+p$, computed at a current instant $k$ using a linear dynamic plant model, and assuming linear inequality constraints, the MPC dynamic optimization problem can be formulated in the following typical quadratic programming (QP) form (see (3.11) in Chapter 3)

$$\begin{aligned}\min_{\triangle u(k|k),\ldots,\triangle u(k+N_u-1|k)} \{ &\sum_{p=N_1}^{N} \|[y^{sp}(k+p|k) - y^0(k+p|k)] + \\ &-\triangle y(k+p|k)\|^2_{\mathbf{\Psi}(p)} + \sum_{p=0}^{N_u-1} \|\triangle u(k+p|k)\|^2_{\mathbf{\Lambda}(p)}\} \\ \text{subj. to: } &\triangle u(k+p|k) \in U_\triangle \\ &u(k+p|k) \in U, \quad p = 0, \ldots, N_u - 1 \\ &y(k+p|k) \in Y, \quad p = N_1, \ldots, N\end{aligned} \tag{4.4}$$

where $\mathbf{\Psi}(p)$ and $\mathbf{\Lambda}(p)$ are diagonal weighting matrices of appropriate dimensions, and constraint sets are assumed of box type (*i.e.*, upper- and lower-bound type, see Section 3.1)

$$U_\triangle = \{\triangle u(k) : \ -\triangle u_{max} \le \triangle u(k) \le \triangle u_{max}\} \tag{4.5}$$

$$U = \{u(k) : \ u_{min} \le u(k) \le u_{max}\} \tag{4.6}$$

$$Y = \{y(k) : \ y_{min} \le y(k) \le y_{max}\} \tag{4.7}$$

Due to linearity of the model, the predicted output trajectory $\{y(k+p|k),\ p = N_1, \ldots, N\}$ is decomposed in (4.4) into a free part $\{y^0(k+p|k),\ p = N_1, \ldots, N\}$ depending on the known current and past signals (calculated with $u(k+p|k) = u(k-1)$ over the prediction horizon), and a forced part $\{\triangle y(k+p|k),\ p = N_1, \ldots, N\}$ depending only on decision variables $\triangle u(k+p|k)$, $p = 0, \ldots, N_u - 1$,

$$y(k+p|k) = y^0(k+p|k) + \triangle y(k+p|k), \quad p = N_1, \ldots, N. \tag{4.8}$$

The set-point trajectory over the prediction horizon, $\{y^{sp}(k+p|k),\ p = N_1, \ldots, N\}$, is directly or indirectly related to the desired optimal steady-state,

as calculated by the steady-state optimization algorithm, either the SSTO or the LSSO, at sampling instants when this information is used. Denoting this steady-state by $y^{ss}$ ($y^{ss} = y^d$, when it stems from LSSO), two typical cases can be distinguished:

- The set-point trajectory is constant over the prediction horizon , then

$$y^{sp}(k+p|k) = y^{ss}, \quad p = N_1, ..., N \tag{4.9}$$

- The set-point trajectory is a reference trajectory (see Section 3.6.2 in Chapter 3). Then it is usually defined as follows

$$y^{sp}(k+p|k) = \gamma y^{sp}(k+p-1|k) + (1-\gamma)y^{ss}, \;\; p = N_1, ..., N \tag{4.10}$$

  where $y^{sp}(k|k) = y(k)$, the value of the process output measured at sampling instant $k$, and $\gamma$ is a design parameter, $0 \leq \gamma < 1$ . This means that a prescribed continuous trajectory approaching the required steady-state, defined by a first-order filter, is applied as a set-point trajectory within the MPC algorithm.

As discussed in the previous section, the best solution would be to always obtain the optimal steady-state from the LSSO problem, as frequently as needed to follow moves of the true optimal steady-state caused by disturbance changes or new input information from the operator. The nonlinear LSSO problem corresponds in the considered case (set-points for a constraint controller are only considered and $\dim c^d = \dim y^d$) to the simplified version of (4.3), having a general form

$$\begin{aligned} &\min Q(c^d, y^d) \\ &\text{subj. to:} \;\; y^d = F(c^d, w) \\ &\qquad\quad c^d \in C^d \\ &\qquad\quad y^d \in Y^d \end{aligned} \tag{4.11}$$

The corresponding MPC-SSTO problems will now be presented, in the order of increasing complexity. Since the SSTO algorithm must run at every sampling instant, before the MPC dynamic optimization, it uses a simpler steady-state model than the LSSO.

### Linear MPC Steady-state Target Optimization

The simplest, standard approach to the MPC-SSTO, applied in industrial practice, see *e.g.*, [62, 142], is based on a linear steady-state process model derived from the dynamic model used in the MPC dynamic optimization, *i.e.*, the gains matrix $\mathbf{G}$ corresponding to this model. Assuming a linear form (4.2) of the economic objective function and linear inequalities (4.6) and (4.7) defining the sets $C^d = U$ and $Y^d$=$Y$, as it occurs in most cases in practice, the discussed *standard SSTO problem* can be formulated in the following LP form

$$\begin{aligned} &\min\{-q^T \triangle y^{ss} + p^T \triangle u^{ss}\} \\ &\text{subj. to: } \triangle y^{ss} = \mathbf{G} \triangle u^{ss} \\ &\qquad y^{ss} = y^0(k+N|k) + \triangle y^{ss} \\ &\qquad u^{ss} = u(k-1) + \triangle u^{ss} \\ &\qquad u_{min} \leq u^{ss} \leq u_{max} \\ &\qquad y_{min} \leq y^{ss} \leq y_{max} \end{aligned} \tag{4.12}$$

where $q$ and $p$ are vectors of prices (see (4.2)), $y^0(k+N|k)$ is the value of predicted free output trajectory at the end of the prediction horizon and $u(k-1)$ is the controller output signal generated at the previous sample.

With the SSTO applied, the following calculations are performed by a MPC constraint controller-optimizer at each sampling instant:

1. Acquisition of the output measurements $y(k)$.
2. Prediction of the free output trajectory to calculate $y^0(k+N|k)$.
3. Solution of the LP SSTO problem (4.12), to get the steady-state targets for the dynamic optimization problem.
4. Solution of the QP dynamic optimization problem (4.4), to obtain the current controller output $u(k)$ as the first element of the calculated optimal control input trajectory.

It should be pointed out that the calculation of the free output prediction $y^0(k+N|k)$ is an important element of the above strategy – this is the only element in the formulations of both the SSTO problem (4.12) and the MPC dynamic optimization problem (4.4) which undergoes significant changes from one sampling instant to the other, as a result of disturbances influencing the output measurements or directly affecting the process model, if measurements of ceratin disturbances are available and explicitly used in the model, see Section 3.2.5 in Chapter 3.

If one wants to require that the steady-state $y^{ss}$ calculated from (4.12) should be achievable over a single prediction horizon using the control generated by the MPC optimization problem (4.4), then the following additional constraint should be added to (4.12)

$$-N_u \cdot \triangle u_{max} \leq \triangle u^{ss} \leq N_u \cdot \triangle u_{max} \tag{4.13}$$

It can happen that for certain values of $u(k-1)$ and $y^0(k+N|k)$ the SSTO problem (4.12) becomes infeasible, a case unacceptable in real-time control. To avoid the infeasibility, a commonly used measure analogous to that applied in MPC dynamic optimization can also be applied here, *i.e.*, recasting the hard output constraints $y_{min} \leq y^{ss} \leq y_{max}$ as soft constraints, by adding slack variables $v_m \geq 0$ and $v_M \geq 0$ that allow for some violation of the constraints (see Section 3.6.2 in Chapter 3). Then the SSTO problem (4.12) becomes

$$
\begin{aligned}
&\min\{-q^T \triangle y^{ss} + p^T \triangle u^{ss} + \rho_1^T v_m + \rho_2^T v_M\} \\
&\quad \text{subj. to:}\ \triangle y^{ss} = \mathbf{G} \triangle u^{ss} \\
&\qquad\qquad y^{ss} = y^0(k+N|k) + \triangle y^{ss} \\
&\qquad\qquad u^{ss} = u(k-1) + \triangle u^{ss} \\
&\qquad\qquad u_{min} \le u^{ss} \le u_{max} \\
&\qquad\qquad -v_m + y_{min} \le y^{ss} \le y_{max} + v_M \\
&\qquad\qquad v_m \ge 0, \quad v_M \ge 0
\end{aligned}
\tag{4.14}
$$

where $\rho_1$ and $\rho_2$ are coefficients of linear penalty terms, penalizing violations of the softened constraints.

In the formulations (4.12) and (4.14) presented so far, the gains matrix $\mathbf{G}$ obtained from the dynamic linear model used in the MPC dynamic optimization problem (4.4) was applied. As discussed previously, this may not be sufficient for more nonlinear processes, since this gains matrix could be then often far from gains corresponding to a current point of the nonlinear steady-state model. A more reasonable option would then be to use a *successive linearization approach*, *i.e.*, to use in SSTO the gains matrix $\mathbf{G}(k)$ calculated from the comprehensive nonlinear model $F(u, w)$ used in the local steady-state optimization (LSSO) problem (4.11), at every sampling instant or after every few sampling instants of the MPC dynamic controller. The form of the linear SSTO problem is then analogous to (4.12):

$$
\begin{aligned}
&\min\{-q^T \triangle y^{ss} + p^T \triangle u^{ss}\} \\
&\quad \text{subj. to:}\ \triangle y^{ss} = \mathbf{G}(k) \triangle u^{ss} \\
&\qquad\qquad y^{ss} = F(u(k-1), w) + \triangle y^{ss} \\
&\qquad\qquad u^{ss} = u(k-1) + \triangle u^{ss} \\
&\qquad\qquad u_{min} \le u^{ss} \le u_{max} \\
&\qquad\qquad y_{min} \le y^{ss} \le y_{max}
\end{aligned}
\tag{4.15}
$$

To avoid possible infeasibility problems, extended formulation analogous to (4.14) can be applied, as well.

Certainly, in the described case the perhaps simplest nonlinear approach, the gain-scheduling approach, to the MPC itself would also be reasonable. That is, the gains of the linear dynamic model used in the MPC output prediction and dynamic optimization should be recalculated at the same sampling instants, to have this model better coinciding with the actual operating point, and the MPC dynamic optimization and MPC SSTO problems consistent.

So far, the problem with the MPC constraint controller having the same number of controlled outputs and control inputs was considered, *i.e.*, it was assumed in (4.4) that $\dim y^{sp} = \dim u$. It is often not the case in practice. The important and interesting case is when there are more control inputs $u$ than

controlled process output variables, $\dim u > \dim y^{sp}$. Then the solution of the MPC optimization problem may be not unique, and the controller should use additional degrees of freedom to force economically better solutions. This could be enforced by adding the following additional constraint to the MPC dynamic optimization problem

$$u(k+N_u-1|k) = u^{ss} \tag{4.16}$$

But a practically better way could be to treat this constraint as a soft one, by formulating the MPC dynamic optimization problem as follows

$$\begin{aligned}
\min_{\triangle u(k|k),\ldots,\triangle u(k+N_u-1|k)} \{ & \sum_{p=N_1}^{N} \|[y^{sp}(k+p|k) - y^0(k+p|k)] + \\
& - \triangle y(k+p|k)\|^2_{\mathbf{\Psi}(p)} + \sum_{p=0}^{N_u-1} \|\triangle u(k+p|k)\|^2_{\mathbf{\Lambda}(p)} + \\
& + \|u^{ss} - u(k+N_u-1|k)\|^2_{\mathbf{R}}\} \\
\text{subj. to: } & -\triangle u_{max} \le \triangle u(k+p|k) \le \triangle u_{max} \\
& u_{min} \le u(k+p|k) \le u_{max}, \quad p = 0,\ldots,N_u-1 \\
& y_{min} \le y(k+p|k) \le y_{max}, \quad p = N_1,\ldots,N
\end{aligned} \tag{4.17}$$

where an additional term with weighting matrix $\mathbf{R}$ is added to the objective function, to force the decision variables to the optimal steady-state values $u^{ss}$ at the end of the control horizon. Having freedom, as $\dim u > \dim y^{sp}$, this does not contradict the basic requirement that steady-state values $y^{ss}$ should be achieved by the controlled outputs at the end of the prediction horizon. It is because the values $u^{ss}$ and $y^{ss}$ are related to each other in the same way as stabilizing values of input and output variables in MPC, provided the same gains matrices are used in both the MPC dynamic optimization and MPC-SSTO problems.

In general, there may also be in the considered case additional measured process outputs entering the constraints only, called the *constraint outputs*, see Section 3.1 and, *e.g.*, [142]. The formulation (4.17) then remains formally unchanged, only a part of the elements of the vector of predicted process outputs enters the performance function, the remaining ones entering the constraints on outputs only.

### Linear-quadratic MPC Steady-state Target Optimization

The successive linearization discussed in the previous point is a kind of adaptive approach, with the aim to adapt the linear model used in the linear SSTO problem to nonlinearity of the problem, to its linearized features at successive operating points. However, one can imagine a representation of nonlinearity in the SSTO problem itself. Then, the first step in this direction would be

to use the *successive quadratic approximation* of the comprehensive nonlinear steady-state process model. The quadratic model can be then written as

$$y^{ss} = F(u(k-1), w) + \mathbf{G}(k) \triangle u^{ss} + 0.5(\triangle u^{ss})^T \mathbf{B}(k) \triangle u^{ss} \tag{4.18}$$

although in general the quadratic term can be recalculated more rare, not at every sampling instant, but, *e.g.*, after every predefined number of sampling instants, or even globally (then $\mathbf{G}(k) = \mathbf{G}$, is a constant matrix) – depending on the kind of nonlinearity of the original steady-state process model.

Now, wanting the SSTO problem be at most linear-quadratic (for quick and robust calculation), *i.e.*, with linear constraints only, it should be formulated as follows

$$\begin{aligned} \min\{-q^T[\mathbf{G}(k) \triangle u^{ss} + 0.5(\triangle u^{ss})^T \mathbf{B}(k) \triangle u^{ss}] + p^T \triangle u^{ss}\} \\ \text{subj. to: } y^{ss} = F(u(k-1), w) + \mathbf{G}(k) \triangle u^{ss} \\ u^{ss} = u(k-1) + \triangle u^{ss} \\ u_{min} \leq u^{ss} \leq u_{max} \\ y_{min} \leq y^{ss} \leq y_{max} \end{aligned} \tag{4.19}$$

Observe that the quadratic approximation is used in (4.19) only in the objective function, because linear approximation must be applied to input-output mappings which describe constrained outputs, to keep the constraints in the optimization problem linear.

Linear approximation of the mappings entering the inequality constraints is the weakest point of this approach, since in many applications optimal solution is constrained (located at constraint limits) and more accurate approximation of the output mappings for the constraints can be more crucial than for the objective function. Despite this weak point, the use of (4.19) can generally lead to better results than the pure linear approach or successive linearization only, especially if only process inputs are constrained. Factors important for the success are nonlinearities well approximated by quadratic functions and design of an efficient quadratic approximation method, if it is to be used at every sampling instant or after every few sampling instants.

It should also be mentioned that there are cases when the original economic-type objective function is not linear, being *e.g.*, quadratic, then the use of a linear-quadratic SSTO problem is natural and recommended.

**Piecewise-linear MPC Steady-state Target Optimization**

The possible weakest point of the just discussed linear-quadratic approach to SSTO was the necessity to use a linear process model in the formulation of the output constraints, as these constraints are often crucial, being active at the optimum. Thus, a question arises of how to design an efficient, robust SSTO problem, but with a nonlinear approach to process modeling. Having

a linear economic objective function, an answer to this question may be a *piecewise-linear approximation* of the nonlinear steady-state process model, performed locally at each sampling instant, after every few sampling instants, or even once, globally – depending on the dimensionality and nonlinearity of the problem. For local approach, the approximation bases on a limited number of points in a current operating region.

*Piecewise-linear approximation* of a nonlinear function is a well-known concept in mathematical programming, especially in separable programming, see *e.g.*, [153]. The idea is explained on an example function approximation in Fig. 4.4. The basis of the approximation is a number of grid points, connected by straight line portions building the approximation, namely points 0(0,0), A$(x_1, y_1)$, B$(x_2, y_2)$ and C$(x_3, y_3)$. One of the ways to do the approximation, called the $\lambda$-*form* [153], is to introduce an additional non-negative variable $\lambda_i$ for each grid point (which value can then be interpreted as a weight of the point in the function approximation), and additional constraints for new variables, as follows:

$$x = 0\lambda_0 + x_1\lambda_1 + x_2\lambda_2 + x_3\lambda_3 \tag{4.20}$$

$$y = 0\lambda_0 + y_1\lambda_1 + y_2\lambda_2 + y_3\lambda_3 \tag{4.21}$$

$$\lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{4.22}$$

$$\lambda_i \geq 0, \ i = 0, 1, 2, 3 \tag{4.23}$$

Moreover, the set of new variables $\{\lambda_0, \lambda_1, \lambda_2, \lambda_3\}$ must be the so-called SOS2 set (special ordered set of type 2), *i.e.*, at most two and adjacent variables can be non-zero. The SOS2 condition can be modeled as an additional set of linear inequalities, especially with 0-1 (integer) additional variables, leading

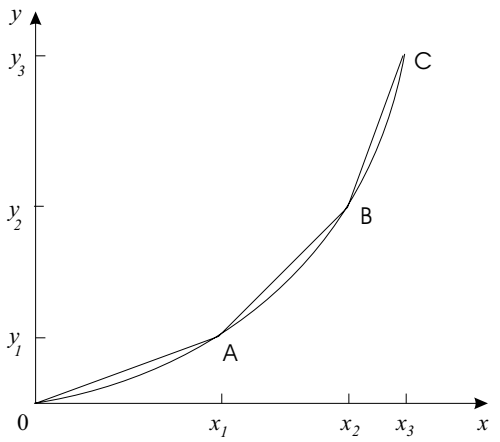


**Fig. 4.4.** An example piecewise-linear approximation of a nonlinear function

to a mixed linear programming problem [153]. But it is not necessary if the applied linear programming procedure allows simply for declarations of SOS2 sets, treating them algorithmically (internally), thus making the modeling easier for the user.

Formulation of a piecewise-linear approximation of a function of two or more variables in a form of a set of linear conditions is more difficult, but can also be made using a number of grid points, introducing additional variables and SOS2 sets. It will not be presented here, the interested reader is referred to *e.g.*, [153]. One of the reasons is that a simpler, more approximate but more practical approach (of a "brute force" type) can also be proposed here and has been found successful in a two-dimensional simulation study, see [138] and Example 4.2 in Section 4.2.4. The basis of this local approach is to use a mesh of grid points in a current operating region and simply to check values of the objective function and nonlinear output constraints at these points only, selecting in this way the best feasible one. Due to linearity of the objective function, the finer, exact solution to the piecewise-linear SSTO problem based on the used mesh points must be located on one of the line fragments connecting the found best point with the unfeasible adjacent ones. But it seems rather not reasonable to look for a finer solution point basing on the used grid mesh (*i.e.*, along the mentioned line fragments), as the problem is itself only an approximate one. If higher accuracy is needed, the search can be repeated with finer grid mesh around the found point, *i.e.*, using a finer piecewise-linear approximation. In fact, the presented approach can be regarded from another point of view: as a simplified, approximate version of the full nonlinear local steady-state optimization (LSSO), performed more frequently and locally, over a smaller region around the operating point.

### 4.2.2 Integrated Approach to MPC and Steady-state Optimization

So far, the need to calculate the steady-state targets for MPC dynamic optimization (MPC-DO) more frequently than it is done at a higher control layer of local steady-state optimization (LSSO) was realized by introduction of a separate SSTO problem, performed at every sampling instant of the MPC controller. Thus, two optimization problems, MPC-DO and MPC-SSTO, were solved sequentially at the MPC constraint control layer. The question arises, whether or not it is possible and reasonable to integrate these two optimization problems into one. Moreover, one can even imagine to integrate the LSSO and the MPC-DO problems into one optimization problem – such attempts can be met in the literature, usually for concrete processes with specific features [145, 158, 159]. Certainly, the problem is difficult, considering its nonlinear nature and requirements of on-line applications.

Therefore, integration of MPC-DO with MPC-SSTO should first be considered. This could be reasonable and lead to faster calculations first of all if structures of both problems are the same or similar, and if the MPC-SSTO

taken as a separate problem is efficacious. Therefore, if this problem formulated as a LP problem (4.12) (or (4.15) if successive linearization is used) is sufficiently efficacious, then it is rather unreasonable to integrate it with the MPC-DO QP problem. Passing further, to more involved MPC target optimization, we can see that (4.4) and (4.19) are both standard QP problems, therefore the following integrated QP problem can be proposed:

$$\begin{aligned}
\min_{\triangle u(k|k),\ldots,\triangle u(k+N_u-1|k),\triangle u^{ss}} \{ & \sum_{p=N_1}^{N} \|[y^{ss} - y^0(k+p|k)] + \\
& - \triangle y(k+p|k)\|^2_{\mathbf{\Psi}(p)} + \sum_{p=0}^{N_u-1} \|\triangle u(k+p|k)\|^2_{\mathbf{\Lambda}(p)} + \\
& + \rho[-q^T[\mathbf{G}(k)\triangle u^{ss} + 0.5(\triangle u^{ss})^T \mathbf{B}(k)\triangle u^{ss}] + p^T \triangle u^{ss}]\}
\end{aligned}$$

$$\begin{aligned}
\text{subj. to: } & -\triangle u_{max} \le \triangle u(k+p|k) \le \triangle u_{max} \\
& u_{min} \le u(k+p|k) \le u_{max}, \quad p = 0, \ldots, N_u - 1 \\
& y_{min} \le y(k+p|k) \le y_{max}, \quad p = N_1, \ldots, N \\
& y^{ss} = F(u(k-1), w) + \mathbf{G}(k)\,\triangle u^{ss} \\
& u^{ss} = u(k-1) + \triangle u^{ss} \\
& u_{min} \le u^{ss} \le u_{max} \\
& y_{min} \le y^{ss} \le y_{max}
\end{aligned} \tag{4.24}$$

where $\rho$ is a weighting coefficient prescribing the relation between the cost function of the MPC-DO problem and the economic objective function. In the above problem, the relation (4.9) between the steady-state value $y^{ss}$ and the set-point trajectory $\{y^{sp}(k+N_1), \ldots, y^{sp}(k+N)\}$ has been assumed (constant set-point over the prediction horizon). The approximation of the nonlinear steady-state model used in the problem (4.24), *i.e.*, matrices $\mathbf{G}(k)$ and $\mathbf{B}(k)$, can be adjusted after every or after a few sampling instants, as required and possible depending on the kind of nonlinearity of the original steady-state process mapping.

Generally, a useful integrated QP problem may also result in other cases when the MPC-SSTO is a QP problem. First of all, when the process model used there is linear, stemming from a linear dynamic one used in MPC-DO, or from linearization of the steady-state nonlinear model, but the original economic objective function is quadratic.

As discussed earlier, one can even imagine integrating MPC-DO and nonlinear LSSO problems into one nonlinear optimization problem. This can be done when the LSSO problem can be effectively and robustly calculated at each sampling instant of the MPC constraint controller, which usually occurs if it is relatively simple and dynamics of the controlled process is sufficiently

slow (*e.g.*, a case considered in [159]). Then, still assuming constant set-point over the prediction horizon, (4.9), the integrated *nonlinear* dynamic optimization problem can be formulated in the following form

$$\begin{aligned}
&\min_{\triangle u(k|k),\ldots,\triangle u(k+N_u-1|k),u^{ss}} \{ \sum_{p=N_1}^{N} \|[y^{ss} - y^0(k+p|k)] + \\
&- \triangle y(k+p|k)\|^2_{\mathbf{\Psi}(p)} + \sum_{p=0}^{N_u-1} \|\triangle u(k+p|k)\|^2_{\mathbf{\Lambda}(p)} + \rho\, Q(u^{ss}, y^{ss})\} \\
&\text{subj. to:} \; -\triangle u_{max} \le \triangle u(k+p|k) \le \triangle u_{max} \qquad (4.25) \\
&\qquad u_{min} \le u(k+p|k) \le u_{max}, \quad p = 0, ..., N_u - 1 \\
&\qquad y_{min} \le y(k+p|k) \le y_{max}, \quad p = N_1, ..., N \\
&\qquad y^{ss} = F(u^{ss}, w) \\
&\qquad u_{min} \le u^{ss} \le u_{max} \\
&\qquad y_{min} \le y^{ss} \le y_{max}
\end{aligned}$$

where $Q(u^{ss}, y^{ss})$ is the economic objective function, which can be fairly general, linear or nonlinear. The properties of (4.25) are usually dominated by the nonlinearity of the steady-state output mapping $F(\cdot, w)$. If this mapping is given by a complex numerical procedure, than a recommended efficacious practical approach is to approximate it by a fuzzy-neural or a neural network model constructed for optimization purposes – to obtain better computational properties of the optimization problem, see [69].

If dim $u^{ss}$ > dim $y^{ss}$, then the constraint (4.16) should also be added to the problem (4.24) or (4.25), as a hard or a soft constraint.

Certainly, the necessity to choose an appropriate value of the weighting coefficient $\rho$ is a drawback of the integrated approach. Too small value will force the solution to be less economically efficient, while too large can lead to undervalued dynamic optimization and numerical difficulties. Of course, a relation between $\rho$ and weighting coefficients being entries of matrices $\mathbf{\Psi}$ and $\mathbf{\Lambda}$ counts, thus one could assume $\rho = 1$ and then choose only entries of these matrices, in general. But, in practice it is usually more convenient to design first the MPC dynamic controller and than the integrated controller, appropriately adjusting the value of $\rho$. The problem will be addressed further on, when presenting and discussing simulation studies.

### 4.2.3 Adaptive MPC Integrated with Steady-state Optimization

An interesting approach to integrated steady-state optimization and MPC dynamic optimization has been proposed in [6], in the framework of adaptive model predictive control with state-space model.

The algorithm requires an instantaneous on-line identification of a linear state-space model, after an initial time interval of several sampling periods needed to start the identification procedure. In fact, the following ARMAX input-output type model is proposed to be identified in [6] (see also (3.82))

$$\mathbf{A}(z^{-1})y(k) = \mathbf{B}(z^{-1})u(k-1) + \mathbf{C}(z^{-1})\epsilon(k) + b \tag{4.26}$$

using a non-iterative moving data window procedure, and the model parameters are updated after every few sampling instants (in the two simulation examples reported it was after 5 and 20 samples, respectively). Then, the following state-space representation is used (see Section 3.4.1)

$$x(k) = [y(k)^T\ y(k-1)^T\ ...\ y(k-n_A+1)^T\ u(k-1)^T\ ...\ u(k-n_B+1)^T]^T \tag{4.27}$$

Formulating the identified linear state-space model in the incremental form and using notation adopted in this book, the *integrated dynamic MPC optimization problem* is as follows

$$\begin{aligned}
\min_{\triangle u(k|k),...,\triangle u(k+N-1|k)} &\{0.5\,\triangle x(k+N|k)^T \mathbf{\Psi}_N \triangle x(k+N|k) + \\
&+ \sum_{p=1}^{N-1} 0.5\,\triangle x(k+p|k)^T \mathbf{\Psi}_p \triangle x(k+p|k) + \\
&+ \sum_{p=0}^{N-1} [0.5\,\triangle u(k+p|k)^T \mathbf{\Lambda}_p \triangle u(k+p|k) + \ Q(u(k+p|k), y(k+p|k))]\}
\end{aligned}$$

$$\begin{aligned}
\text{subj. to: } & \triangle x(k+p+1|k) = \mathbf{A}(k)\,\triangle x(k+p|k) + \mathbf{B}(k)\,\triangle u(k+p|k) \\
& \triangle y(k+p|k) = \mathbf{C}(k)\,\triangle x(k+p|k) \\
& -\triangle u_{max} \le \triangle u(k+p|k) \le \triangle u_{max} \\
& u_{min} \le u(k+p|k) \le u_{max}, \quad p = 0, ..., N-1 \\
& y_{min} \le y(k+p|k) \le y_{max}, \quad p = 1, ..., N
\end{aligned} \tag{4.28}$$

where $Q(u, y)$ is the economic objective function and $\mathbf{\Psi_N}$, $\mathbf{\Psi}_p$, $\mathbf{\Lambda}_p$ are weighting matrices. Observe that in (4.28) the economic objective function is taken into account at every sampling instant over the prediction horizon and is not multiplied by a weighting coefficient. Observe also that the control horizon is assumed to be equal to the prediction horizon.

Note that, as usual with adaptive systems based on current model identification, the process input signal must have sufficient persistent excitation properties to get acceptable robustness and quality of the identification. Therefore, in simulation studies presented in [6], an additional PRBS (pseudo-random binary sequence) signal added to the controller output was applied.

A proof of the optimality of a process equilibrium point (steady-state $(\bar{y}, \bar{u})$) obtained under the presented integrated algorithm is given in [6], under several assumptions. Not going into more technical ones (the reader is

referred to the cited paper for more detailed information), let us only mention the more important assumptions:

- The prediction horizon $N$ is infinite.
- Matrices $\mathbf{A}(k)$, $\mathbf{B}(k)$ and $\mathbf{C}(k)$ of the adaptive linear state-space model stabilize (at values, say, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$) when approaching the steady-state under stabilizing (constant) disturbances, *i.e.*, the adaptive control system is assumed to be asymptotically stable.
- Derivative of the stabilized linear model is equal to the derivative of the real steady-state input/output process mapping $y = F_*(u, w)$ at the equilibrium point $(\bar{y}, \bar{u})$, *i.e.*,

$$\mathbf{C}(\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \left[\frac{\partial F^*(\bar{u}, w)}{\partial u}\right] \tag{4.29}$$

Under the assumed conditions, it is proved in [6] that the steady-state point $(\bar{y}, \bar{u})$ satisfies necessary optimality conditions of the *real process* steady-state optimization problem

$$\begin{aligned} &\min Q(u,y) \\ &\text{subj. to: } y = F_*(u,w) \\ &\qquad\qquad u \in U \\ &\qquad\qquad y \in Y \end{aligned} \tag{4.30}$$

Note that the problem (4.30) is the steady-state optimization problem (4.1), but with the real input-output process mapping $F_*$ in place of its model $F$. Certainly, the real mapping $F_*$ cannot be known in practice, only its (approximate) model $F$ is known, therefore the knowledge of $F_*$ cannot be assumed. But, assuming (4.29), it is possible to attain the true optimal point (precisely, a point satisfying necessary optimality conditions of the true optimum). And this assumption is more realistic, although not easy to be met in practice. It means that steady-state properties of *every* state-space linear dynamic model identified during the operation of the algorithm must coincide with the true input-output steady-state process mapping at the point of identification, when considering the first order derivatives. It is difficult in practice, as linear dynamic models are usually identified to obtain a good match of dynamic properties of the modeled process. To obtain good static properties, special measures must be taken, like appropriate low-pass filtering of input and output signals used for identification purposes.

The problem (4.30) will be called the steady-state optimizing control problem in Section 4.3.1, to distinguish from the steady-state model optimization problem. In fact, a quite fortunate combination of the philosophy of integrated system optimization and parameter estimation (ISOPE) approach, with the MPC technique, is behind the construction of the discussed algorithm. The ISOPE methodology will be discussed briefly in Section 4.3.1.

### 4.2.4 Comparative Example Results

**Example 4.1**
Control of a jacketed continuously-stirred tank reactor with polymerization reaction described in [34, 84] and considered earlier in Example 3.8 in Chapter 3, see Fig. 3.46, will now be considered. But now, our interest will be the structure consisting not only of a feedback MPC control, but containing also a combined set-point optimization. Different realizations of SSTO together with different frequencies of intervention of the LSSO will be considered [70].

Differential equations describing the process are almost the same, with the initiator inflow rate as the control input $u = F_I$, but now the main disturbance input, the monomer and solvent feed flow rate $F$ will not be constant. Therefore the set of difference equations is now (compare with Example 3.8)

$$\begin{aligned}
\dot{x}_1 &= 10(6 - x_1) - 2.4568x_1\sqrt{x_2} \\
\dot{x}_2 &= 80u - (0.1022 + 10F)x_2 \\
\dot{x}_3 &= 0.0024121x_1\sqrt{x_2} + 0.112191x_2 - 10x_3 \\
\dot{x}_4 &= 245.978x_1\sqrt{x_2} - 10x_4 \\
y &= x_4/x_3
\end{aligned}$$

The nominal operating conditions are the same as stated in Example 3.8, with the same constraints on the control input $u$. The predictive controller applied to keep the process at desired set-points was the GPC controller, see also Example 3.8. However, it is not important now which MPC controller is applied, provided it operates well for the considered range of the set-point values. The output variable $y$ is NAMW (number-average molecular weight), controlled by manipulating the inlet initiator flow rate $F_I$ ($u = F_I$).

The controlled process behavior will now be driven by changes in the disturbance input

$$F(k) = F_0 - 0.5(sin(0.0145k) - sin(0.145))$$

where $F_0 = 1$ $[m^3/h]$. The economic objective is to minimize the NAMW, $Q(c, y) = y$.

Three different control system structures were considered, with the following realizations of the on-line set-point optimization task:

- LSSO – with nonlinear steady-state optimization at the optimization layer only,
- LSSO+SSTO – with LSSO and steady-state target optimization (SSTO), in three versions (see Section 4.2.1):
  - LSSO+SSTOLc – SSTO with a constant gain matrix $\mathbf{G}$ (linear static model) from the linear dynamic model used in the GPC algorithm, see (4.12),

  - LSSO+SSTOLa – SSTO with an adaptive gain matrix $\mathbf{G}(k)$ (linear static model) from on-line successive linearizations of the comprehensive nonlinear steady-state model used at the optimization layer (in LSSO), see (4.15),
  - LSSO+SSTOQ – SSTO with a quadratic static model from on-line successive approximations of the nonlinear steady-state model used in LSSO, see (4.19),
- LSSO+SSTOQiMPC – with LSSO and integrated optimization problem combining SSTO and GPC dynamic optimization, as described in Section 4.2.2.

Selected simulation results obtained with the presented control-optimization structures are shown in Figures 4.5, 4.6, 4.7 and 4.8, for intervention period of the comprehensive economic optimization (LSSO) $T_{LSSO} = 30$ $(30T_p)$.

In Figures 4.5 and 4.6 results obtained in control-optimization structures LSSO, LSSO+SSTOLc and LSSO+SSTOLa are shown. It can clearly be seen that the introduction of the SSTO leads to significant improvement, and results obtained for adaptive static process gain stemming from on-line linearizations of the nonlinear steady-state model (LSSO+SSTOLa structure) were, for the considered plant, almost identical to those with a constant gain; only a slight difference between process input trajectories can be seen.

In Figures 4.7 and 4.8 the results obtained in control-optimization structures LSSO, LSSO+SSTOQ and LSSO+SSTOQiMPC are shown. The application of SSTO with quadratic approximations yielded in the considered case results with performance slightly worse than that obtained for linear models. The application of an integrated approach was also investigated. It improves the results when compared with the LSSO case only, but they are worse than those for separate SSTO with quadratic approximation. It should be remembered that in the integrated case an additional separate optimization (SSTO) is not needed.

The influence of different values of $T_{LSSO}$ on the economic performance was also investigated. The results, calculated for the whole simulation interval, are shown in Figure 4.9, for the applied control-optimization structures (since both cases of SSTO with linear models led to almost the same results, only one is depicted). The best result, as expected, was for the (rather not realistic) case with LSSO repeated at every sampling instant of the GPC controller, *i.e.*, for $T_{LSSO} = 1$. To get the presentation clearer, relative values $J_{rel}$ of the objective function are shown, related to this best result with $T_{LSSO} = 1$, *i.e.*,

$$J_{rel} = (J - J_{T_{LSSO}=1}) \cdot 10^{-4}$$

For $T_{LSSO}$ significantly larger than 1, using the LSSO only leads to bad results. It is evident, that application of the SSTO working with sampling frequency of the MPC algorithm is important, and best practical results are for combined LSSO+SSTO structure. With the increase of $T_{LSSO}$ the results deteriorate in the considered problem, however slower for larger intervention periods.

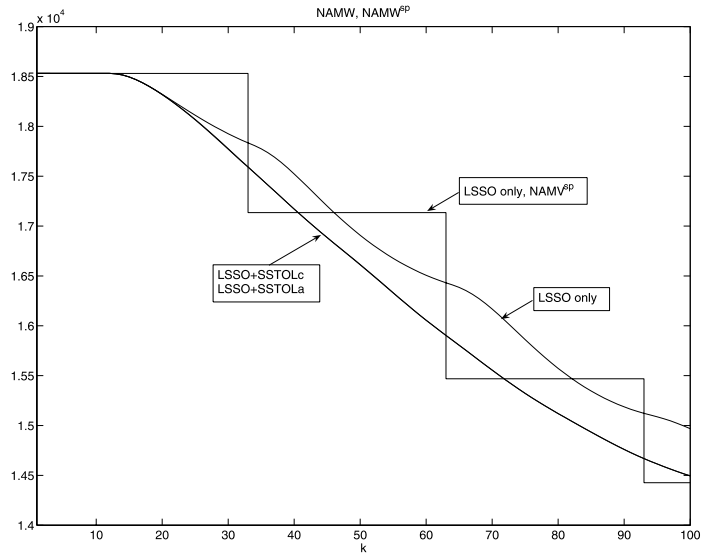


**Fig. 4.5.** Trajectories of NAMW for structures without and with SSTO implemented as LP problem

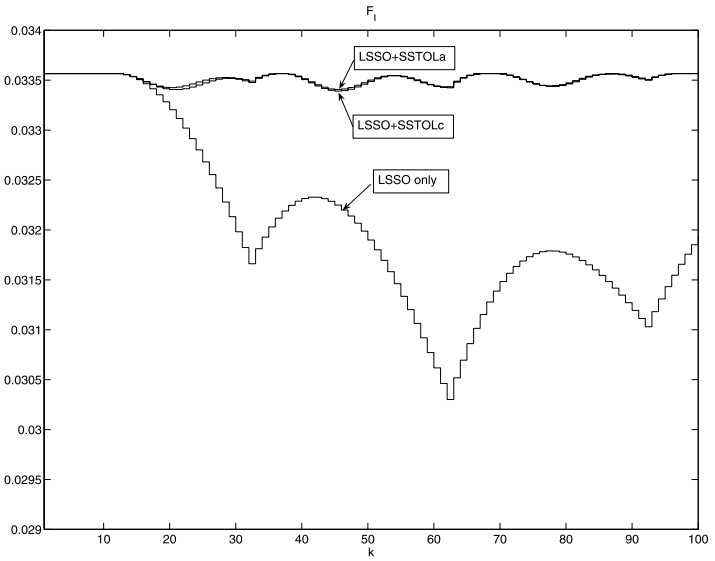


**Fig. 4.6.** Trajectories of process control input for structures without and with SSTO implemented as LP problem

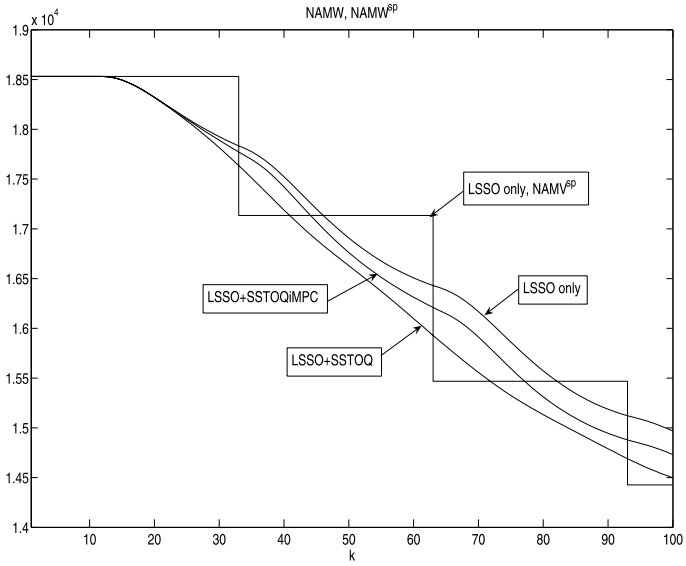


**Fig. 4.7.** Trajectories of NAMW for structures without and with SSTO implemented as QP problem, including integrated approach

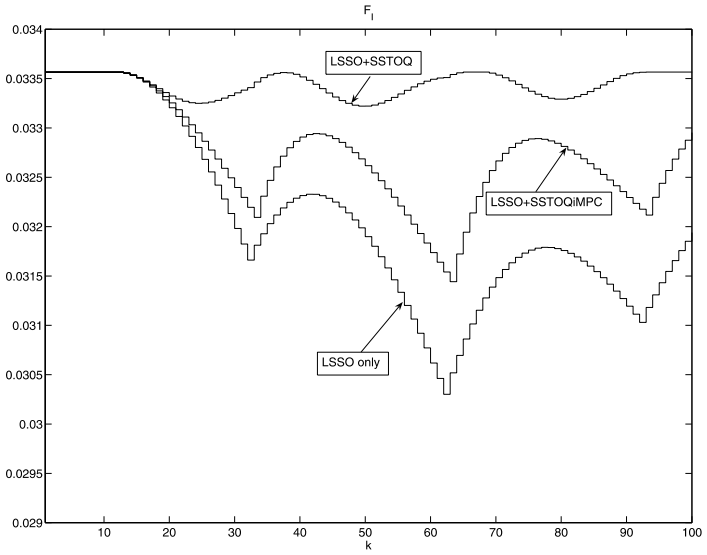


**Fig. 4.8.** Trajectories of process control input for structures without and with SSTO implemented as QP problem, including integrated approach

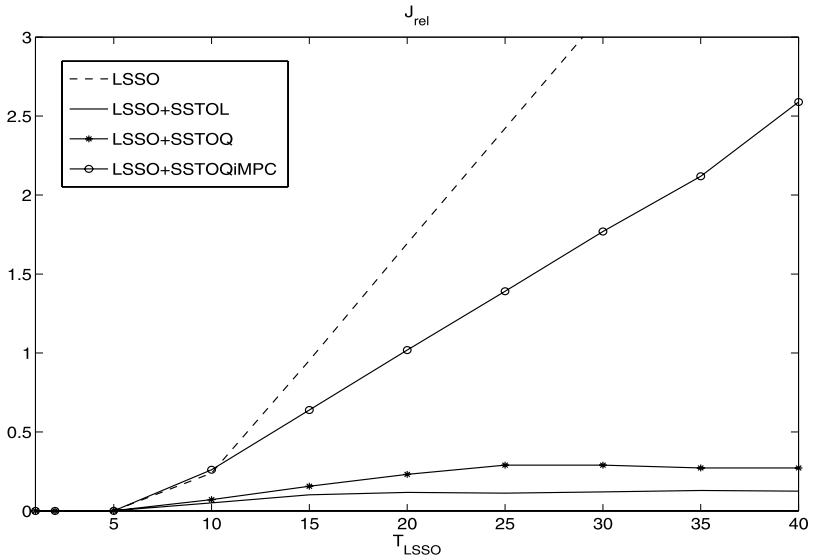


**Fig. 4.9.** Dependence of the loss of the relative performance on the intervention period of the LSSO optimization

The considered example problem has a rather smooth, regularly changing and not very strong nonlinearity of the steady-state model. The obtained results show, that in this case the application of the SSTO with a linear model leads to good results, even using a linear model with a constant gain matrix. This confirms practical experience, where in many cases this well-known simple approach can be successful, see *e.g.*, [115, 11]. However, the linear approach can not be sufficient in cases with sharply changing nonlinearities, as it will be shown in the next example. □

**Example 4.2**

A methanol-water distillation column described earlier in Example 2.3 in Chapter 2 will be considered, but now with the MPC constraint controller added, as it is shown in Fig. 4.10. The top product consists mainly of methanol (95%), whereas the bottom product contains only traces of alcohol (less than 5%). The feed flow rate is denoted by $F$, the flow rates of the top and bottom products are denoted by $D$ and $B$, respectively. Concentrations of methanol in the feed, top and bottom streams, are denoted by $x_f$, $x_d$ and $x_b$, respectively. The plant has two manipulated variables: $R$ - reflux stream flow rate and $V$ - vapor stream flow rate. The operating point is: $R_0 = 33.3634$ $[kmol/h]$, $V_0 = 83.3636$ $[kmol/h]$, $x_{d0} = 0.95$, $x_{b0} = 0.05$ and $F_0 = 100$ $[kmol/h]$, $x_{f0} = 0.5$ (compositions are in mole fractions).

Two fast single-loop PID controllers, denoted as LC, are used to maintain the levels in reflux and bottom product tanks. Similar controllers, denoted as FC, are used to control the actual flow rates of $R$ and $V$. The direct

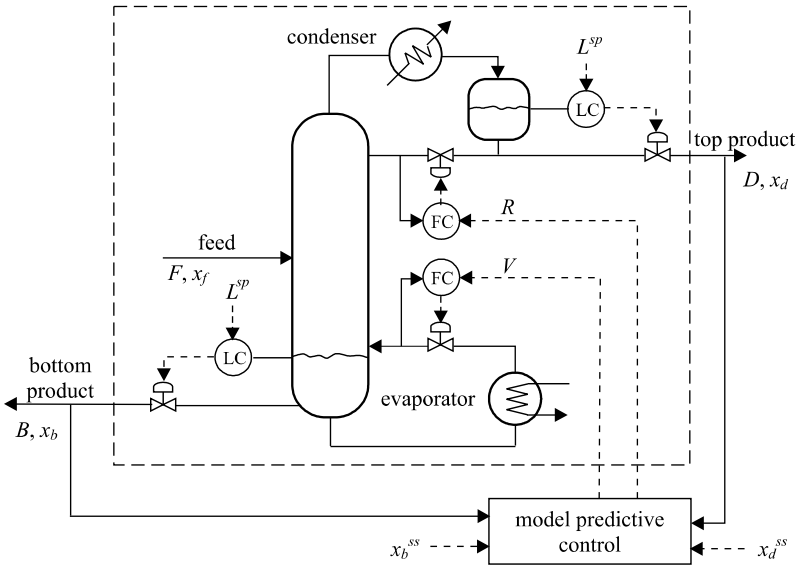


**Fig. 4.10.** Methanol-water distillation column feedback control structure

control layer comprises all LC and FC controllers. In order to stabilize the concentrations $x_d$ and $x_b$ in top and bottom products, a supervisory MPC constraint controller is used. It treats the column as a two-input ($R$, $V$) two-output ($x_d$, $x_b$) process, the parameters of the input stream ($F$, $x_f$) are disturbances. The sampling interval of this algorithm is equal to 10 $[min]$.

A GPC realization was chosen for the MPC controller, with prediction and control horizons set to $N = 10$ and $N_u = 5$ , with weighting matrices $\mathbf{\Psi}$=diag$\{10, 5\}$ and $\mathbf{\Lambda} = \lambda\mathbf{I}$, where $\lambda = 1$. In the GPC dynamic optimization problem the following amplitude and rate of change constraints on the input variables were imposed (for $p = 0, ..., N_u - 1$):

$$R_{min} \leq R(k+p|k) \leq R_{max}$$

$$V_{min} \leq V(k+p|k) \leq V_{max}$$

$$-\triangle R_{max} \leq \triangle R(k+p|k) \leq \triangle R_{max}$$

$$-\triangle V_{max} \leq \triangle V(k+p|k) \leq \triangle V_{max}$$

where $R_{min} = R_0 - 20$, $R_{max} = R_0 + 20$, $V_{min} = V_0 - 20$, $V_{max} = V_0 + 20$, $\triangle R_{max} = 5$, $\triangle V_{max} = 5$ (all flow rates in $[kmol/h]$). It was also assumed that $F = F_0$ and that changes in feed stream composition $x_f$ (main disturbance) are given by the following equation (for $k \geq 10$)

$$x_f(k) = x_{f0} + 0.1(\sin(0.029k) - \sin(0.29))$$

The steady-state (set-point) optimization problem was formulated as follows

$$\begin{aligned} \min\{-c_D D^{ss} + c_B B^{ss} + c_R R^{ss} + c_V V^{ss}\} \\ \text{subj. to: } R_{min} \le R^{ss} \le R_{max} \\ V_{min} \le V^{ss} \le V_{max} \\ x_d^{ss} \ge 0.95 \\ x_b^{ss} \le 0.05 \end{aligned} \tag{4.31}$$

where $c_D$=2.5, $c_B$=0.5, $c_R$=1 and $c_V$=1 are appropriate prices. The last two constraints in the optimization problem (4.31) result from technological requirements: the top-product has to be sufficiently purified whereas the bottom product has to contain only traces of methanol.

The first-principles (fundamental) dynamic model was used as the process representation during the simulations [81]. It was also deployed to generate the data and to carry out an identification procedure of a dynamic black-box linear model to be used in the GPC algorithm. In fact, the objective function in (4.31) is linear, since at steady-states bottom and top product flows result from balance equations

$$\begin{aligned} D^{ss} &= V^{ss} - R^{ss} \\ B^{ss} &= R^{ss} + F^{ss} - V^{ss} \end{aligned}$$

However, to calculate the concentrations $x_d^{ss}$ and $x_b^{ss}$, present in the last two inequality constraints in (4.31), a first-principles static model has to be used,

$$\begin{aligned} x_d^{ss} &= f^d(R^{ss}, V^{ss}, x_f^{ss}, F^{ss}) \\ x_b^{ss} &= f^b(R^{ss}, V^{ss}, x_f^{ss}, F^{ss}) \end{aligned}$$

Therefore, the optimization problem is nonlinear.

Results of applications of LSSO combined with SSTO, in the presented control structure, were originally presented in [138]. First, the SSTO with a linear model was considered. The obtained results, see Fig. 4.11, were far from satisfactory, due to the strong nonlinear shape of the steady-state characteristics. The operating points obtained were in each iteration close to $x_d^{ss}$=0.95 and $x_b^{ss}$=0, whereas the optimal set-point for the process is located close to $x_d^{ss}$=0.95 and $x_b^{ss}$=0.05 (as it was checked using the nonlinear model). Unfortunately, the same problem occurred when SSTO with successive linearizations was applied, and the SSTO with quadratic approximations could not be used because the nonlinearity is located in constraints, see Section 4.2.1.

Therefore, a SSTO with piecewise-linear approximations was tried, see Section 4.2.1. Because dimensionality of the problem is small, it can be relatively quickly solved using a straightforward procedure which calculates values of the objective function and checks the fulfilment of the nonlinear constraints for a predefined set (a mesh) of possible steady-state points – certainly, with

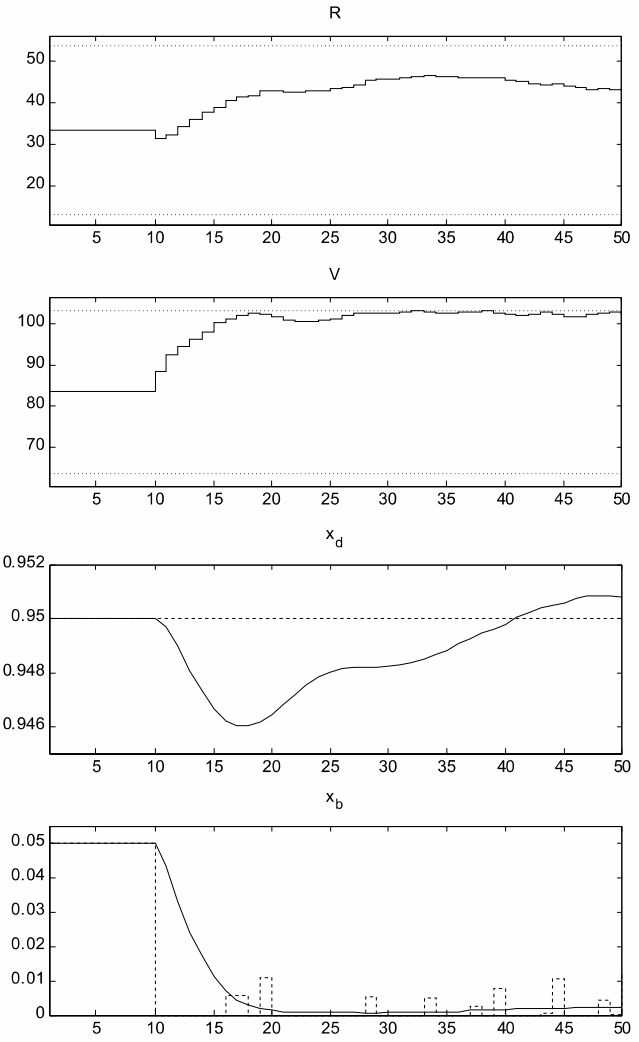


**Fig. 4.11.** Trajectories of process inputs and set-points obtained for SSTO with linear process model only

the accuracy corresponding to differences between function values at neighboring mesh points. During simulations a mesh consisting of only 25 candidate grid points was used. The last steady-state operating point determined by the nonlinear optimization layer (LSSO) was the central point of the mesh. The obtained results, see Fig. 4.12, led to much better values of the objective function than those received using the linear approach [138].

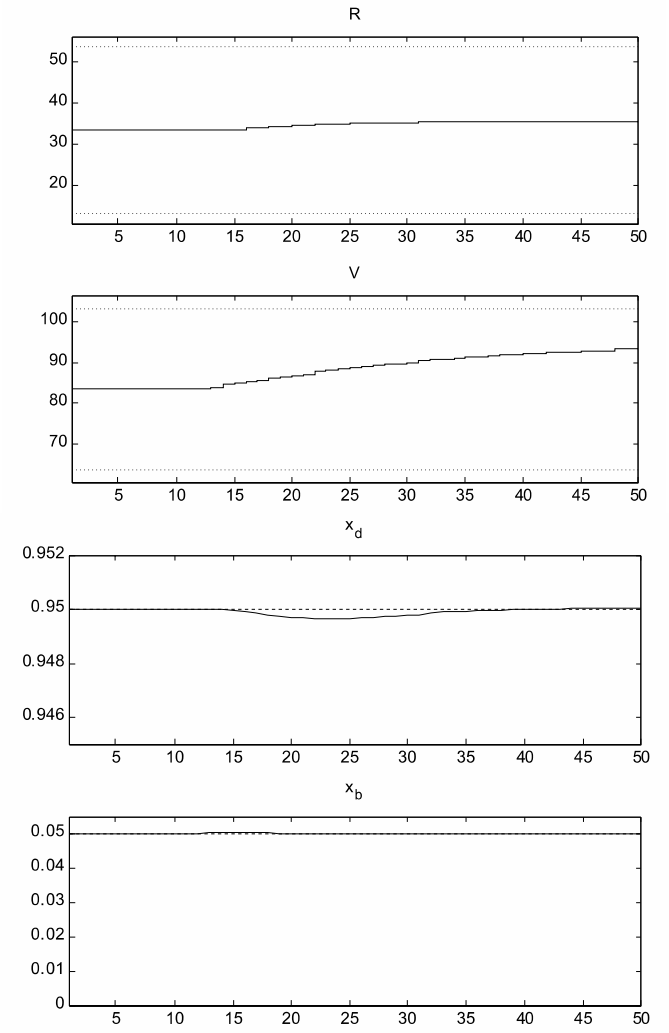


**Fig. 4.12.** Trajectories of process inputs and set-points obtained for SSTO with piecewise-linear process model □

## 4.3 Measurement-based Iterative Set-point Optimization under Uncertainty

In a classical multilayer structure it is usually assumed that disturbances considered at the optimization layer are varying slowly or in an abrupt manner but rarely, when compared to the controlled process dynamics, see *e.g.*, [40, 16]. Then the steady-state optimization problem can be solved with a frequency

much lower than the sampling frequency of directly subordinate feedback controllers; the time between two successive optimizations can be much longer than the duration of step-response transient processes in the controlled plant. This case will be assumed in this section.

From the point of view of the steady-state optimization, the disturbances $w$ are parameters. To have the model of the process $F$ describing the reality well, values of these disturbances should be known – measured or estimated. Only then the solution of the steady-state optimization problem with the process model does determine the operating point (set-point) which is optimal for the real process. More precisely, it is then close enough to the real optimal point. It is not strictly optimal due to unavoidable uncertainties in disturbance measurement or estimation and inaccuracies in modeling. Otherwise, the point evaluated using an (only rough) process model can be far from the optimal one for the real process. But, the situation that the uncertainty, although unknown, can be assumed constant over longer periods of time (much longer than the controlled process settling time) can be exploited for iterative improvement of the model-optimal point, based on successive use of steady-state measurement information. This leads to iterative methods, the most important being methods of integrated system optimization and parameter estimation (ISOPE), see [16].

### 4.3.1 Integrated System Optimization and Parameter Estimation (ISOPE)[1]

The ISOPE method was proposed by Roberts [121], see also [122], it was originally called a *modified two-step method*. However, it gained popularity under the name ISOPE, that is now usually used. The feature that makes the method so attractive is that it is able to generate a series of set-points *converging to the plant real optimal steady-state point* (not only model-optimal point), in spite of the uncertainty in the process model and disturbance estimates.

To explain the idea of the method and the basic ISOPE algorithm, we shall consider in this section the case without constraints on process outputs, leaving this more complex case to the next section. The constraints on process inputs will be described in a general compact form

$$c \in C = \{c : g(c) \leq 0\} \tag{4.32}$$

Further on, because disturbances can be assumed to be constant during a single run of any ISOPE algorithm and the process model is assumed to be rough, it will be defined in the form with certain model parameters $\alpha$ representing the uncertainty, namely

[1]This section is a shortened and modified version of Sections 4.1 and 4.4 from the book Brdys, M.A. and Tatjewski, P., *Iterative Algorithms for Multilayer Optimizing Control*, copyright 2005 by Imperial College Press, used by permission.

$$y = F(c, \alpha) \tag{4.33}$$

Now, using the introduced assumptions and notation, let us write the steady-state *model optimization problem* (MOP) (4.1) in the form

$$\begin{aligned} &\min_{c} Q(c, y) \\ &\text{subj. to: } y = F(c, \alpha) \\ &\qquad\qquad g(c) \leq 0 \end{aligned} \tag{4.34}$$

and the steady-state *optimizing control problem* (OCP) (4.30) as

$$\begin{aligned} &\min_{c} Q(c, y) \\ &\text{subj. to: } y = F_*(c, w) \\ &\qquad\qquad g(c) \leq 0 \end{aligned} \tag{4.35}$$

with the optimal solution (optimal steady-state point for the real process) denoted by $\hat{c}_*$.

To derive the ISOPE basic algorithm, let us convert the OCP problem (4.35) to an equivalent form with the output variables $y$ eliminated

$$\begin{aligned} &\min_{c,\alpha} Q(c, F(c, \alpha)) \\ &\text{subj. to: } F(c, \alpha) = F_*(c, w) \\ &\qquad\qquad g(c) \leq 0 \end{aligned} \tag{4.36}$$

and next, still preserving full equivalence, to the form with additional variables $c^i$ representing a current steady-state point (at which parameter adaptation will be performed, the superscript "$i$" will index algorithm iterations)

$$\begin{aligned} &\min_{c,c^i,\alpha} \{Q(c, F(c, \alpha)) + \rho \|c^i - c\|^2\} \\ &\text{subj. to: } F(c^i, \alpha) = F_*(c^i, w) \\ &\qquad\qquad g(c) \leq 0 \\ &\qquad\qquad c^i = c \end{aligned} \tag{4.37}$$

where the last constraint has been added to maintain equivalence and a convexifying quadratic term has also been added to the objective function $Q(c, F(c, \alpha))$. The augmented problem is, certainly, equivalent, and the augmentation leads to better applicability conditions of the resulting algorithm, see [13, 16], and provides the designer with a useful additional parameter $\rho$ (penalty coefficient).

Writing now the Lagrange function for the problem (4.37),

$$\begin{aligned} L(c^i, c, \alpha, \lambda, \xi, \mu) \;=\; & q(c, \alpha) + \rho \|c^i - c\|^2 + \\ & + \lambda^T (c^i - c) + \xi^T (F(c^i, \alpha) - F_*(c^i, w)) + \mu^T g(c) \end{aligned} \tag{4.38}$$

where

$$q(c,\alpha) = Q(c, F(c,\alpha)) \tag{4.39}$$

the necessary optimality conditions for (4.37) can be written as:

$$q_c^{'}(c,\alpha)^T - 2\rho(c^i - c) - \lambda + g^{'}(c)^T\mu = 0 \tag{4.40a}$$

$$g(c) \le 0, \quad \mu \ge 0, \quad \mu^T g(c) = 0 \tag{4.40b}$$

$$2\rho(c^i - c) + \lambda + [F_c^{'}(c^i,\alpha) - (F_*)_c^{'}(c^i,w)]^T\xi = 0 \tag{4.40c}$$

$$q_\alpha^{'}(c,\alpha)^T + F_\alpha^{'}(c^i,\alpha)^T\xi = 0 \tag{4.40d}$$

$$F(c^i,\alpha) - F_*(c^i,w) = 0 \tag{4.40e}$$

$$c^i - c = 0. \tag{4.40f}$$

Eliminating now from (4.40c) and (4.40d) the Lagrange multipliers $\xi$, the following equivalent formulation of these necessary optimality conditions is obtained

$$q_c^{'}(c,\alpha)^T - \lambda(c^i,\alpha) - 2\rho(c^i - c) + g^{'}(c)^T\mu = 0 \tag{4.41a}$$

$$g(c) \le 0, \quad \mu \ge 0, \quad \mu^T g(c) = 0 \tag{4.41b}$$

$$F(c^i,\alpha) - F_*(c^i,w) = 0 \tag{4.41c}$$

$$c^i - c = 0 \tag{4.41d}$$

where

$$\lambda(c^i,\alpha)^T = Q_y^{'}(c^i, F(c^i,\alpha)) \cdot [F_c^{'}(c^i,\alpha) - (F_*)_c^{'}(c^i,w)] \tag{4.42}$$

For a full, rigorous derivation and a more elaborate discussion the reader is referred to [16].

The ISOPE algorithms can be treated as those finding, in an iterative way, a point satisfying the necessary optimality conditions (4.41a)–(4.42). First, let us assume that also parameters $\alpha$ are iterated, writing $\alpha^i$ instead of $\alpha$, and define the *modified model optimization problem* MMOP (compare with the model optimization problem MOP, (4.34)):

$$\begin{aligned} &\min_c \{q(c,\alpha^i) - \lambda(c^i,\alpha^i)^T c + \rho\|c^i - c\|^2\} \\ &\text{subj. to:} \;\; g(c) \le 0 \end{aligned} \tag{4.43}$$

Notice that (4.41a)–(4.41b) with $\alpha = \alpha^i$ are precisely the necessary optimality conditions for the MMOP (4.43).

We shall not provide the reader with a full analysis of applicability conditions of the ISOPE method, it can be found elsewhere, see *e.g.*, [16] and references therein. Let us only mention that an important assumption is that the model $F(\cdot,\cdot)$ should be *point parametric* on its feasibility set $C$, see [12, 16], *i.e.*,

$$\text{for every } \bar{c} \in C \text{ there is } \bar{\alpha} \in \mathbb{R}^s \text{ such that } F_*(\bar{c}, w) = F(\bar{c}, \bar{\alpha}) \tag{4.44}$$

The above assumption can be regarded as a standard requirement for the model to be well-defined. It states that it is possible to make the model output equal to the plant output, at any point in the feasible set associated with the original problem (4.35), by an appropriate choice of the parameters $\alpha$. The assumption is, *e.g.*, always satisfied when the model is additive with respect to a subset, say $\alpha_a$, of the parameters, *i.e.*,

$$F(c, \alpha) = F(c, \alpha_n) + \alpha_a, \quad \alpha = (\alpha_n, \alpha_a), \quad \alpha_a \in \mathbb{R}^m \tag{4.45}$$

where $m = \dim y$.

The *parameter estimation problem* (PEP) will be defined as a problem of adapting the model parameters $\alpha$ at an operating point (set-point) $c$ under the constraint (4.41c)

$$F(c^i, \alpha) - F_*(c^i, w) = 0 \tag{4.46}$$

PEP is well-defined, for every $c^i \in C$, if the model is point-parametric on $C$. In particular, if the model has the structure (4.45), then condition (4.46) can always be satisfied by adapting only the additive parameters $\alpha_a$, in fact by a simple substitution

$$\alpha_a^i = F_*(c^i, w) - F(c^i, \alpha_n) \tag{4.47}$$

Generally, there are also non-additive parameters $\alpha_n$, and if both subsets of parameters are adapted the PEP may have not a unique solution. However, as it will be clear from the presentation to follow, it is not necessary to adjust the non-additive parameters $\alpha_n$ at every iteration of the ISOPE (at every PEP solution). These parameters may even be kept constant during a single run of the algorithm, if it is reasonable, *i.e.*, if there is not sufficient additional measurement information gathered to perform new full parameter adaptation during a single run.

We are now ready to formulate the ISOPE algorithms.

**ISOPE Basic Algorithm**

*The ISOPE basic algorithm* can be formulated as follows:

Start. Given initial point $c^0$, relaxation coefficient $k_c$, $0 < k_c \leq 1$, and solution accuracy $\epsilon > 0$. Set $i = 0$.

Step 1. Apply $c^i$ as the controlled plant set-point and measure $y^i = F_*(c^i, w)$, in the steady-state. Perform additional linearly independent perturbations around $c^i$ and measure corresponding values of the plant outputs, in steady-states. Based on this measurements find a finite difference approximation of the process input-output mapping derivative $(F_*)_c'(c^i, w)$.

Step 2. Using the obtained new measurements update parameters $\alpha$ under the restriction that the model outputs match the actual process outputs at $c^i$ (*parameter estimation problem*). This yields $\alpha^i = \hat{\alpha}(c^i)$ satisfying

$$y^i = F(c^i, \alpha^i) = F_*(c^i, w) \tag{4.48}$$

Step 3. For $\alpha = \alpha^i$ solve the *modified model optimization problem* MMOP (4.43), let $\hat{c}_m^i = \hat{c}_m(c^i, \alpha^i)$ be the solution.

Step 4. Set

$$c^{i+1} = c^i + k_c(\hat{c}_m^i - c^i) \tag{4.49}$$

If

$$\|c^i - c^{i+1}\| \leq \epsilon \tag{4.50}$$

then terminate (equality (4.41d) satisfied – solution found). If not, then increase $i$ by one and continue from Step 1.

It can easily be seen that the algorithm is constructed in such a way that when it terminates, the necessary optimality conditions (4.41a)–(4.41d) for the optimizing control problem are satisfied. Equalities (4.41a), (4.41b) and (4.41c) are fulfilled at each iteration after solving the PEP and MMOP problems in Steps 2 and 3. When the algorithm terminates equation (4.41d) is satisfied with the prescribed accuracy $\epsilon > 0$.

The presented ISOPE algorithm can be regarded as of fix point type, since the set-points $c$ are iterated in such a way as to fulfill the equation (4.41d), which in the algorithm realization takes the form

$$\hat{c}_m(c, \hat{\alpha}(c)) = c \tag{4.51}$$

The iterative formula (4.49) is a simple adjustment rule for finding a fix point of (4.51); it is called the iteration of a relaxation type and the parameter $k_c$ is called the relaxation coefficient. Observe that if $k_c = 1$ then this formula becomes a direct substitution rule $c^{i+1} = \hat{c}_m^i$. The formula (4.49) possesses a very important practical property: if the feasible set $C$ is convex, the initial point $c^0 \in C$ and $0 < k_c \leq 1$, then $c^1 \in C$ and, consequently, each point $c^i$ of the generated sequence is feasible (belongs to $C$).

Full theoretical analysis of optimality and convergence properties of the presented algorithm can be found elsewhere [13, 16]. Let us only mention that it has been proven, under reasonable technical conditions, that

- If a point $c^i$ from the generated sequence does not satisfy the optimality conditions, then

$$Q(c^{i+1}, F_*(c^{i+1}, w)) < \ Q(c^i, F_*(c^i, w)) \tag{4.52}$$

  for every $k_c$ from a certain interval, $0 < k_{c\,min} \leq k_c \leq k_{c\,max} \leq 1$ (values $k_{c\,min}$ and $k_{c\,max}$ being problem dependent), *i.e.*, the value of the process real (not model-based) objective function is improved.
- There is at least one cluster point of the sequence $\{c^i\}$ and this point satisfies (4.51), *i.e.*, it satisfies necessary optimality conditions of the optimizing control problem (OCP).

It should be pointed out that at each iteration of the ISOPE algorithm the real process mapping derivative $(F_*)_c'(c^i, w)$ must be evaluated (approximated), in order to compute the modifier $\lambda$, see (4.42), which is necessary for

the MMOP formulation. This derivative must be evaluated based on process measurements only, locally at a current set-point, since the model of the process mapping is assumed to be too uncertain to be used for that. The way this derivative is evaluated is one of the key-points of each ISOPE algorithm, important for its practical features and effectiveness. In the basic ISOPE algorithm structure given above, this derivative is calculated in Step 1 using finite difference approximations based on output measurements from $n$, $n = \dim c$, additional set-point perturbations around the current value $c^i$. This is the simplest approach originally suggested in [121] and utilized in many later papers, see *e.g.*, [122, 14, 13, 139]. However, it means $n$ additional transient processes in the plant at each iteration of the ISOPE algorithm, in addition to the single transient process associated with the set-point transition from $c^{i-1}$ to $c^i$. Therefore, this procedure is highly time and cost expensive.

Therefore, attempts have been made to overcome this drawback of the ISOPE approach, to find more time-effective ways of calculating the derivative $(F_*)'_c(c^i, w)$. In [160] it is proposed to identify first a linear dynamic model of the controlled process around each iteration point $c^i$, actively from a sequence of dynamic measurements resulting from a sequence of perturbations around the current set-point. Then, the obtained linear dynamic model is used to calculate the required derivative of the steady-state process mapping, in a way explained in Section 4.2.3, when this approach has been incorporated into the integration of the basic idea to the ISOPE with an adaptive MPC algorithm. The discussed active dynamic identification approach is an alternative, but has a number of practical drawbacks, as discussed in Section 4.2.3.

In [15] a novel ISOPE *dual algorithm* was proposed, based also on *active* approach to derivative approximation, but applied only to the generated sequence of steady-states (set-points) $c^i$. This should be, when compared to the previous dynamic identification approach, clearly advantageous, as measurement noise can be much easier filtered out from steady-state measurements.

At each iteration of the dual algorithm, the next set-point $c^{i+1}$ is generated in such a way that not only the ISOPE task to optimize the economic objective function is performed, but also, simultaneously, it is assured that $c^{i+1}$ is appropriately located for the purpose of future derivative approximation. This may lead to some loss of current optimality, but at the same time it anticipates future measurement needs. This is, obviously, a dual structure in the sense of control duality, as defined in the control theory. Therefore, the resulting algorithm is called the *ISOPE Dual (ISOPED)* algorithm.

### ISOPE Dual Algorithm

The derivative approximation is based on the collection of last n+1 set-points $c^i, c^{i-1}, \ldots, c^{i-n}$ which must be such that all vectors

$$s^{ik} = c^{i-k} - c^i, \quad k = 1, ..., n \tag{4.53}$$

are linearly independent. Then the following matrix $S^i$,

$$S^i = [c^{i-1} - c^i \quad c^{i-2} - c^i \quad \cdots \quad c^{i-n} - c^i]^T \tag{4.54}$$

is nonsingular.

If the $j$-th plant output mapping $F_{*j}$ has continuous partial derivatives in a neighborhood of $c^i$, then its directional derivative $DF_{*j}(c^i, w; s^{ik})$ at a point $c^i$ and in a direction $s^{ik} = c^{i-k} - c^i$ can be computed as

$$DF_{*j}(c^i, w; s^{ik}) = \frac{1}{\|s^{ik}\|}(s^{ik})^T \nabla_c F_{*j}(c^i, w) \tag{4.55}$$

which can be written in the form

$$\left\|s^{ik}\right\| DF_{*j}(c^i, w; s^{ik}) = (s^{ik})^T \nabla_c F_{*j}(c^i, w). \tag{4.56}$$

Writing the above $n$ equations, *i.e.*, for $n$ directions $s^{ik}$, $k = 1, \ldots, n$, in a matrix form we get

$$S^i \nabla_c F_{*j}(c^i, w) = \begin{bmatrix} \left\|s^{i1}\right\| DF_{*j}(c^i, w; s^{i1}) \\ \vdots \\ \left\|s^{in}\right\| DF_{*j}(c^i, w; s^{in}) \end{bmatrix}, \quad j = 1, \ldots, m. \tag{4.57}$$

If the points $c^{i-k}$ are close enough to $c^i$, then the following approximation can be applied

$$DF_{*j}(c^i, w; s^{ik}) \cong \frac{F_{*j}(c^i + s^{ik}, w) - F_{*j}(c^i, w)}{\|s^{ik}\|} \tag{4.58}$$

The set of equations (4.57) can now be expressed in the approximate form

$$S^i \nabla_c F_{*j}(c^i, w) \cong \begin{bmatrix} F_{*j}(c^{i-1}, w) - F_{*j}(c^i, w) \\ \vdots \\ F_{*j}(c^{i-n}, w) - F_{*j}(c^i, w) \end{bmatrix}, \quad j = 1, \ldots, m \tag{4.59}$$

In the ISOPED algorithm the consecutive points $c^i$ are generated in such a way that the derivative estimation (4.59) of the process input-output mapping can be effectively applied, at each iteration.

Calculation of the required derivative approximations from (4.59) will be practically useful only if the matrix $S^i$ is not only nonsingular, but also *sufficiently well-conditioned*. The reason is that the right-hand side of (4.59) is corrupted not only by the method error (approximation of the right-hand side of (4.57)), but also by plant output measurement errors. Good conditioning can be achieved only by enforcing appropriate location of the consecutive set-points in $\mathbb{R}^n$. It is achieved in the dual ISOPE algorithms by introducing a new inequality constraint to the modified model optimization problem MMOP, see (4.43), called *the conditioning constraint*, in the form

$$d(c^{i+1}(c), c^i, \ldots, c^{i-n+1}) = \frac{\sigma_{min}(S^{i+1}(c))}{\sigma_{max}(S^{i+1}(c))} \geq \delta \tag{4.60}$$

where

$$c^{i+1}(c) = c^i + k_c(c - c^i) \tag{4.61}$$

$$S^{i+1}(c) = [c^i - c^{i+1}(c) \quad \cdots \quad c^{i-n+1} - c^{i+1}(c)]^T \tag{4.62}$$

and $\sigma_{min}(S^{i+1}(c))$, $\sigma_{max}(S^{i+1}(c))$ denote minimal and maximal singular values of $S^{i+1}(c)$. The value of $\delta$, $0 < \delta < 1$, defines the required *conditioning* of the matrix $S^{i+1}(c)$. Notice that $\frac{\sigma_{min}(S^{i+1}(c))}{\sigma_{max}(S^{i+1}(c))}$ is the reciprocal of the standard condition number of the matrix $S^{i+1}(c)$.

The *ISOPE Dual algorithm* (ISOPED algorithm) can now be formulated:

Start. Given initial point $c^0$, parameter of the conditioning constraint $\delta > 0$, relaxation coefficient $k_c$, $0 < k_c \leq 1$, solution accuracy $\epsilon > 0$.

Step 0 (initial phase). Set $c^{-n} = c^0$. Choose $n$ points $c^{-n+1}, c^{-n+2}, \ldots, c^0$ such that the matrix $S^0$ is sufficiently well conditioned. Apply, consecutively, $c^i$ as the controlled plant set-points and measure $F_*(c^i, w)$, $i = -n+1, -n+2, \ldots, -1$. Set $i = 0$.

Step 1. Apply $c^i$ as the controlled plant set-point and measure $F_*(c^i, w)$, in the steady-state. Calculate $(F_*)_c'(c^i, w)$ according to (4.59), *i.e.*, solve $m$ sets of $n$ linear equations

$$S^i \cdot (F_{*j})_c'(c^i, w)^T = \begin{bmatrix} F_{*j}(c^{i-1}, w) - F_{*j}(c^i, w) \\ \vdots \\ F_{*j}(c^{i-n}, w) - F_{*j}(c^i, w) \end{bmatrix}, \quad j = 1, \ldots, m. \tag{4.63}$$

Step 2. Using the obtained measurements update the parameters $\alpha$ of the model $F$ under the restriction to match the process and model outputs at $c^i$ (*parameter estimation problem*). This yields $\alpha^i = \hat{\alpha}(c^i)$ satisfying

$$y^i = F(c^i, \alpha^i) = F_*(c^i, w) \tag{4.64}$$

Step 3. Solve the *modified model optimization problem* MMOP (4.43) denoting its solution by $\hat{c}^i_{m0}$.

Step 4. Set $c_0^{i+1} = c^i + k_c(\hat{c}^i_{m0} - c^i)$. If

$$\|c_0^{i+1} - c^i\| \leq \epsilon \tag{4.65}$$

then terminate (solution found).

Step 5. If

$$d(c_0^{i+1}, c^i, \ldots, c^{i-n+1}) \geq \delta \tag{4.66}$$

then set $c^{i+1} = c_0^{i+1}$, increase $i$ by one and continue from Step 1.
If not, then solve the *conditioned modified model optimization problem* (CMMOP) defined as follows:

$$\begin{aligned} &\min_c \{Q(c, F(c, \alpha^i)) - \lambda(c^i, \alpha^i)^T c + \rho\|c^i - c\|^2\} \\ &\text{subj. to:} \quad c \in C \cap D^i \end{aligned} \tag{4.67}$$

where

$$D^i = \{c \in \mathbb{R}^n : d(c^{i+1}(c), c^i, \ldots, c^{i-n+1}) \geq \delta\} \tag{4.68}$$

and $c^{i+1}(c)$ is given by (4.61). Denote the solution by $\hat{c}^i_m$.

Step 6. Set

$$c^{i+1} = c^i + k_c(\hat{c}^i_m - c^i) \tag{4.69}$$

increase $i$ by one and continue from Step 1.

The first four steps of the presented ISOPED algorithm correspond to the ISOPE basic algorithm, with the addition of an initial phase and different way of derivative approximation. The initial task in Step 5 is to check if $c_0^{i+1}$ satisfies the additional conditioning constraint. If it does, then it can be taken as the next set-point (for the next algorithm iteration), without any change in the basic algorithm and, therefore, without any current loss of optimality. If it does not, then it is suitably modified to satisfy the conditioning constraint. This constraint reduces the feasible set of the modified model optimization problem. Hence, it might be that a current loss of optimality will be observed. However, the new set-point generated in this way anticipates approximation requirements of the next iteration, which in turn should lead to better optimality of the set-point generated next. This can be described as active gathering of suitable measurement information, incorporated into the optimizing control algorithm, thus resulting in a dual-goal algorithm.

For analysis of the presented algorithm, in particular of the properties of the conditioning set, the reader is referred to [16]. The more efficient version with optimized initial phase (optimized procedure to perform Step 0) can be also found there.

### Simulation Results

Extensive simulation results of control systems with both ISOPE basic and dual algorithms, the latter with standard and optimized initial phase, for a few problems can be found in [16]. We shall present here only selected results for a distillation column example.

**Example 4.3** [2]

A distillation column depicted in Fig. 4.13 will be considered, where ethylene is distilled from a mixture containing mainly ethane and ethylene.

The output vector $y = [y_1 \ y_2]^T$ represents values (in steady-states) of concentrations of ethane in the product stream $P$, measured in *ppm* (*parts per million*) and of ethylene in the bottom product $W$, given as a molar fraction. The vector of decision variables $c = [c_1 \ c_2]^T$ represents set-points for feedback controllers stabilization:

[2] Shortened version of Example 4.4 from the book Brdys, M.A. and Tatjewski, P., *Iterative Algorithms for Multilayer Optimizing Control*, copyright 2005 by Imperial College Press, used by permission.

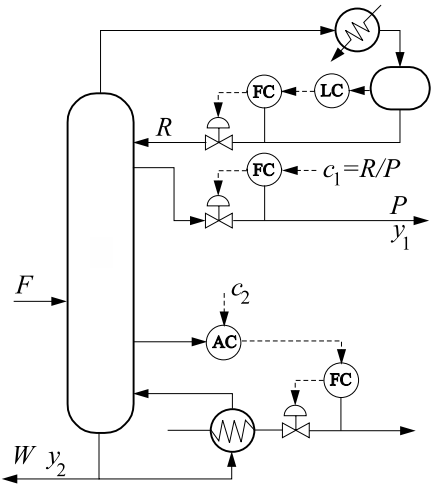


**Fig. 4.13.** Distillation column for Example 4.3 (reproduced from Brdys, M.A. and Tatjewski, P., *Iterative Algorithms for Multilayer Optimizing Control*, page 117, copyright 2005 by Imperial College Press, used by permission)

$c_1$ – ratio of reflux flow $R$ to product flow $P$ (forced by the manipulation of $P$ with respect to measured value of $R$),

$c_2$ – ethylene concentration on the control shelf.

Models of the column which were considered during simulations have been taken from [130], where on the basis of input-output data records two models have been developed: a "real plant" model for simulation purposes and a simplified model for control purposes. The real plant model will serve as a real plant mapping during the simulations, it is of the form

$$y_1 = F_{*1}(c_1, c_2) = \exp(-s_{11}(c_1 - \bar{c}_{11})) \cdot \exp(-s_{12}(c_2 - \bar{c}_{12})) \tag{4.70a}$$

$$y_2 = F_{*2}(c_1, c_2) = \exp(-s_{21}(c_1 - \bar{c}_{21})) \cdot \exp(-s_{22}(c_2 - \bar{c}_{22})) \tag{4.70b}$$

where

$$\begin{aligned} s_{11} &= 12.7049, \quad \bar{c}_{11} = 4.6816 \\ s_{12} &= 0.2536, \quad \bar{c}_{12} = 0.3252 \\ s_{21} &= 0.3340, \quad \bar{c}_{21} = 2.5544 \\ s_{22} &= -5.3719, \quad \bar{c}_{22} = 1.1838. \end{aligned}$$

The simplified model (for control purposes) is of the form

$$y_1 = F_1(c_1, c_2, a) = m_1(c_1 - \bar{c}_{11m})^4 \cdot (c_2 - \bar{c}_{12m})^4 + \alpha_{a1} \tag{4.71a}$$

$$y_2 = F_2(c_1, c_2, a) = m_{21}c_1 + m_{22}c_2 + \alpha_{a2} \tag{4.71b}$$

where

$$m_1 = 999.9923, \quad \bar{c}_{11m} = 4.6261, \quad \bar{c}_{12m} = 2.6481, \quad \alpha_{a1} = 0$$
$$m_{21} = -0.0016, \quad m_{22} = 0.0254, \quad \alpha_{a2} = 0.0043$$

and $\alpha_a = [\alpha_{a1}, \alpha_{a2}]$ is the vector of additive model parameters. These parameters will be updated when performing a parameter estimation (PEP) during runs of the ISOPE algorithms. It is assumed that it is enough to update other (non-additive) model parameters $\alpha_n = (m_1, \bar{c}_{11m}, \bar{c}_{12m}, m_{21}, m_{22})$ before initialization of the ISOPE algorithm and that they are kept constant during its single run.

The goal of the set-point optimization is to achieve desired steady-state values $y_{r1}$ and $y_{r2}$ of the plant outputs. Therefore, the objective function is formulated as

$$Q(c, y) = Q(y) = \mu_1(y_1 - y_{r1})^2 + \mu_2(y_2 - y_{r2})^2 \tag{4.72}$$

where $\mu_1 = 2 \cdot 10^{-5}$ and $\mu_2 = 10^6$ are scaling coefficients needed due to very different scales of the outputs $y_1$ (hundreds) and $y_2$ (in the range to $10 \times 10^{-3}$). Typical desired set-point values are $y_{r1} = 500$, $y_{r2} = 0.005$, and these values were assumed in the simulations. The set-points $c \in \mathbb{R}^2$ are constrained to the set

$$C = \{\, c \in \mathbb{R}^2 : \;\; 4.1 \le c_1 \le 4.6, \quad 0.2 \le c_2 \le 0.4 \,\} \tag{4.73}$$

The presented objective function (4.72) has an indirect economical sense. Namely, the more polluted the product the cheaper its production (less energy consumed for distillation). Therefore, it is optimal to run the distillation at the desired value $y_{r1}$ (ethane concentration), located as close as possible to the largest admissible value, but at a distance of a safety zone of a width resulting from the uncertainty level in the control system. On the other hand, losses of ethylene should not be too high in the bottom product — therefore, a soft constraint on its concentration in this product, expressed as a desired value $y_{r2}$.

First, the objective function (4.72) was optimized subject to the model constraints (4.71a), (4.71b) and inequality constraints, yielding the model-optimal point $(\hat{c}_{m1},\ \hat{c}_{m2}) = (4.2685,\ 0.2964)$. The corresponding values of the plant mapping (4.70a), (4.70b) are $F_*(\hat{c}_{m1},\ \hat{c}_{m2}) = [191.6 \;\; 0.0048]^T$ – and are obviously far from the desired values $y_{r1} = 500$, $y_{r2} = 0.005$, especially from the first one. Then, simulations of the optimizing control using the ISOPED algorithm were performed. The conditioned modified model optimization problem (CMMOP) takes for the presented problem the form

$$\begin{aligned} &\min_c \{Q(F(c, \alpha^i)) - \lambda(c^i, \alpha^i)^T c + \rho \|c^i - c\|^2\} \\ &\text{subj. to:} \;\; 4.1 \le c_1 \le 4.6 \\ &\qquad\qquad 0.2 \le c_2 \le 0.4 \\ &\qquad\qquad d(c^{i+1}(c), c^i, c^{i-1}) = \frac{\sigma_{\min}(\mathbf{S}^{i+1}(c))}{\sigma_{\max}(\mathbf{S}^{i+1}(c))} \ge \delta \end{aligned} \tag{4.74}$$

where $c = [c_1\ c_2]^T$, $c^{i+1}(c) = c^i + k_c(c - c^i)$ and

$$\mathbf{S}^{i+1}(c) = \begin{bmatrix} c_1^i - c_1^{i+1}(c) & c_1^{i-1} - c_1^{i+1}(c) \\ c_2^i - c_2^{i+1}(c) & c_2^{i-1} - c_2^{i+1}(c) \end{bmatrix}^T \tag{4.75}$$

Further, in the formulation of the CMMOP we have :

$$Q(F(c, \alpha^i)) = \mu_1(F_1(c, \alpha^i) - y_{r1})^2 + \mu_2(F_2(c, \alpha^i) - y_{r2})^2 \tag{4.76}$$

$$\lambda(c^i, \alpha^i)^T = [2\mu_1(F_1(c^i, \alpha^i) - y_{r1})\ \ 2\mu_2(F_2(c^i, \alpha^i) - y_{r2})] \times$$
$$\times \left( \begin{bmatrix} \frac{\partial F_1}{\partial c_1}(c^i, \alpha^i) & \frac{\partial F_1}{\partial c_2}(c^i, \alpha^i) \\ \frac{\partial F_2}{\partial c_1}(c^i, \alpha^i) & \frac{\partial F_2}{\partial c_2}(c^i, \alpha^i) \end{bmatrix} - \begin{bmatrix} \frac{\partial F_{*1}}{\partial c_1}(c^i) & \frac{\partial F_{*1}}{\partial c_2}(c^i) \\ \frac{\partial F_{*2}}{\partial c_1}(c^i) & \frac{\partial F_{*2}}{\partial c_2}(c^i) \end{bmatrix} \right) \tag{4.77}$$

where

$$\alpha_1^i = F_{*1}(c^i) - m_1(c_1^i - \bar{c}_{11m})^4 \cdot (c_2^i - \bar{c}_{12m})^4 \tag{4.78}$$

$$\alpha_2^i = F_{*2}(c^i) - m_{21}c_1^i + m_{22}c_2^i \tag{4.79}$$

because only the additive parameters were assumed to be updated during the runs of the ISOPED algorithm. The partial derivatives of the model mapping needed for (4.77) are calculated directly from the formulae (4.71a) i (4.71b), whereas vectors of the "real plant" input-output mapping derivatives

$$(F_{*j})'(c^i) = \begin{bmatrix} \frac{\partial F_{*j}}{\partial c_1}(c^i) & \frac{\partial F_{*j}}{\partial c_2}(c^i) \end{bmatrix}, \quad j = 1, 2$$

are calculated, during the runs of the algorithm, from solutions of linear equations

$$\mathbf{S}^i \begin{bmatrix} \frac{\partial F_{*j}}{\partial c_1}(c^i) \\ \frac{\partial F_{*j}}{\partial c_2}(c^i) \end{bmatrix} = \begin{bmatrix} F_{*j}(c^{i-1}) - F_{*j}(c^i) \\ F_{*j}(c^{i-2}) - F_{*j}(c^i) \end{bmatrix}, \quad j = 1, 2 \tag{4.80}$$

where

$$\mathbf{S}^i = \begin{bmatrix} c^{i-1} - c^i & c^{i-2} - c^i \end{bmatrix}^T \tag{4.81}$$

Simulations of the optimizing control using the ISOPED algorithm were performed for the presented plant description and its model. The results shown in Figures 4.14 and 4.15 present trajectories generated by the algorithm starting from the set-point equal to the model-optimal one, $(\hat{c}_{m1},\ \hat{c}_{m2}) = (4.2685,\ 0.2964)$, *i.e.*, from the solution to the MOP problem. The following algorithm parameters were assumed: $\rho = 100$, $k_c = 1$, $\epsilon = 0.001$. A relatively small value of $\epsilon$ was assumed intentionally, to investigate convergence in a small neighborhood of the optimum (in practical optimizing control applications this value should be larger, adjusted to output measurement errors

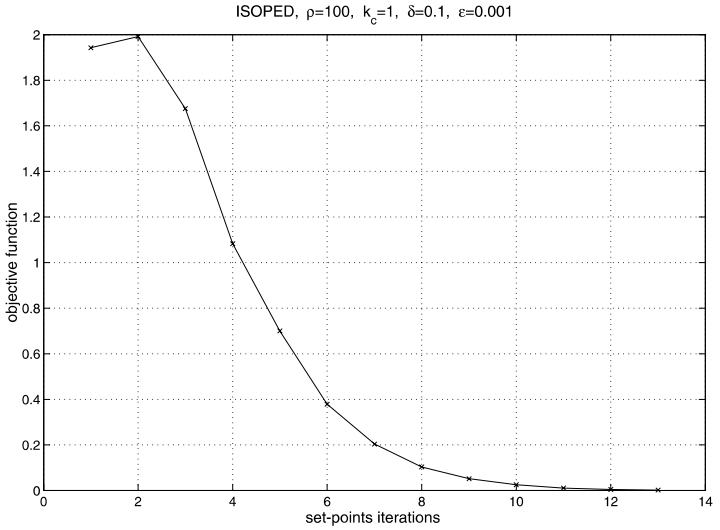


**Fig. 4.14.** Real objective function trajectory (reproduced with modifications from Brdys, M.A. and Tatjewski, P., *Iterative Algorithms for Multilayer Optimizing Control*, page 122, copyright 2005 by Imperial College Press, used by permission)

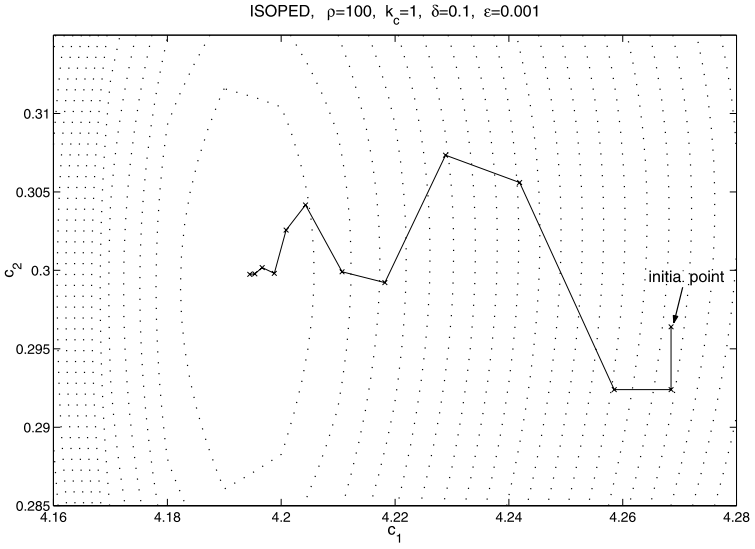


**Fig. 4.15.** Trajectory of set-points (reproduced with modifications from Brdys, M.A. and Tatjewski, P., *Iterative Algorithms for Multilayer Optimizing Control*, page 122, copyright 2005 by Imperial College Press, used by permission)

and differences). In Fig. 4.14 trajectories of the real objective function value $Q(c^i, F_*(c^i))$ are presented, whereas in Fig. 4.15 trajectories of the decision variables (set-points), plotted against a background consisting of contour lines (dotted lines) of the performance function $Q(c, F_*(c))$. In the latter figure the initial phase (Step 0) of the dual algorithm can well be seen, consisting of two first set-point changes in directions of the space versors.

Further simulation results, in particular for other initial points and when using ISOPED algorithm with optimized initial phase, can be found in [16]. □

### 4.3.2 ISOPE for Problems with Output Constraints

In complex processing plants usually many output variables are constrained and constraint controllers (see Fig. 4.2) are designed to keep the important constraints satisfied, as discussed in Chapter 1 and in the introductory part of this chapter. Set-points for the constraint controllers are the desired steady-state values of constrained, feedback controlled process outputs, see Fig. 1.3 in Chapter 1, where the vector of these process outputs was denoted by $y^d$ – being, in general, a sub-vector of the overall output vector $y = (y^f, y^d)$. The vector of the decision variables $c$ of the optimization problem was similarly there divided into two parts, $c = (c^f, c^d)$. The first sub-vector represents set-points for direct controllers, while the second one represents optimal steady-state values of variables corresponding to outputs of constraint controllers – see also Fig. 1.3 – both sub-vectors calculated on the basis of the steady-state process model used at the optimization unit, including the available information about current values of disturbances $w$.

We shall assume a constant structure of the constraint controller during every single run of the ISOPE algorithm, *i.e.*, that the same outputs are constrained at the same required values $y^d_r$, see Fig. 1.3. In practice, an even more stringent situation often takes place, when the same vital constraints on process outputs are required to be satisfied over longer periods of time, *e.g.*, required concentrations in product streams.

To explain in a simple way how the ISOPE method can be applied in a structure with constraint controllers, we shall also consider the simplified case with feedback constrained outputs only, *i.e.*, $y = y^d$, as this does not lead to loss of generality.

The steady-state *model optimization problem* (4.3) then takes the following simplified form, with model parameters $\alpha$ representing parametric uncertainty, as in the previous section,

$$\begin{aligned} &\min_{c^f, c^d} Q(c^f, c^d, y^d) \\ &\text{subj. to:}\ \ y^d = F(c^f, c^d, \alpha) \\ &\qquad\qquad y^d = y^d_r \\ &\qquad\qquad g(c^f) \le 0 \end{aligned} \tag{4.82}$$

Inequality constraints on free set-points $c^f$ have only been assumed in (4.82). This means, in practice, that the control system designer has managed to select such components $c^d$ out of the vector $c$ that do not enter their constraint limits during the process operation. This does not always occur in practice, but simplifies the presentation to follow.

The *optimizing control problem* (OCP) corresponding to the model optimization problem (4.82) is the following

$$\begin{aligned} &\min_{c^f, c^d} Q(c^f, c^d, y^d) \\ &\text{subj. to: } y^d = F_*(c^f, c^d, w) \\ &\qquad\qquad y^d = y^d_r \\ &\qquad\qquad g(c^f) \leq 0 \end{aligned} \tag{4.83}$$

Applying a given constant value of $c^f$ to the controlled system results in a corresponding steady-state, in a certain steady-state value of the constraint controller output, and thus in a corresponding steady-state value of $c^d = c^d_r + \delta c^d$, if the structure of Fig. 1.3 with the constraint controller acting as a "correction controller" is applied. Therefore, due to the feedback action of the constraint controller, from the point of view of the steady-state optimizer, the set-points $c^d$ are in fact dependent variables, fully dependent on $c^f$ (which is emphasized by the superscript $d$). This functional relation between the steady-state values of $c^d$ and $c^f$ will be denoted by $P_*$,

$$c^d = P_*(c^f, y^d_r, w) \tag{4.84}$$

Observe that $P_*$ is a mapping implicitly hidden in the equality

$$F_*(c^f, c^d, w) = y^d_r \tag{4.85}$$

and the vector of dependent set-points $c^d$ is usually chosen in such a way that $P_*$ is a well-defined mapping. This will be assumed in this section, implying also that the constraint controller operates well, in a unique way.

Using the introduced mapping (4.84), the OCP problem (4.83) can be transformed to the following form

$$\begin{aligned} &\min_{c^f} Q(c^f, c^d, y^d_r) \\ &\text{subj. to: } c^d = P_*(c^f, y^d_r, w) \\ &\qquad\qquad g(c^f) \leq 0 \end{aligned} \tag{4.86}$$

The mapping $P_*$ is certainly unknown, as stemming from the real process input-output mapping. It is possible to build a model of $P_*$ from the data only, but an alternative is to build first a model of the input-output process mapping

$$y^d = F(c^f, c^d, \alpha) \tag{4.87}$$

where $\alpha$ are adjustable model parameters, see (4.82). A model $P$ of $P_*$ can then be obtained as a solution, with respect to $c^d$, of the implicit system of equations

$$F(c^f, c^d, \alpha) = y_r^d \tag{4.88}$$

Certainly, in more involved cases the explicit form of $P$ may be difficult to derive, then the set of equations (4.88) can be treated as the required model in an implicit numerical form, yielding for every given "input" $c^f$ the "output" $c^d$ calculated numerically, as a solution to the set of equations. Further, after collecting a number of such input-output data a simplified explicit form of the model $P$ can be constructed.

The model optimization problem (MOP) corresponding to the optimizing control problem (4.86) is

$$\begin{aligned} &\min_{c^f} Q(c^f, c^d, y_r^d) \\ &\text{subj. to: } c^d = P(c^f, y_r^d, \alpha) \\ &\qquad\qquad g(c^f) \leq 0 \end{aligned} \tag{4.89}$$

Comparing the steady-state optimizing control and model optimization problems (4.35) and (4.34) considered in the previous section with (4.86) and (4.89), it can easily be seen that formulations of both pairs of problems have precisely the same structure, only in the latter pair there is $P_*$ and $P$ in place of $F_*$ and $F$, and $c^f$ in place of $c$ ($y_r^d$ being an additional constant parameter). Thus, the derivation and formulation of ISOPE algorithms presented in Section 4.3.1 can be directly applied here, with the mentioned obvious differences. Therefore, it will not be presented here. For a full, detailed presentation, in the general case with both unconstrained and feedback constrained process outputs, the interested reader is referred to [135, 16]. Results of an example application to a complex styrene production plant consisting of three distillation columns working in series are also presented there.

Formulations of ISOPE algorithms for problems with inequality constrained process outputs in control structures without dynamic feedback constraint controllers are also possible. These constraints must then be treated algorithmically, leading to more involved ISOPE algorithms. The reader is referred to [16] for an explanation and further references.

# References


1. F. Allgöwer, T.A. Badgwell, J.S. Qin, J.B. Rawlings, and S.J. Wright. Nonlinear predictive control and moving horizon estimation – an introductory overview. In P.M. Frank, editor, *Advances in Control – Highlights of ECC'99*, chapter 12. Springer, London, 1999.
2. K.J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, Reading, Massachusetts, 1995.
3. K.J. Åström and B. Wittenmark. *Computer Controlled Systems*. Prentice Hall, Upper Saddle River, 1997.
4. R. Babuska, J.M. Sousa, and H.B. Verbruggen. Predictive control of nonlinear systems based on fuzzy and neural models. In *Proc. 5th European Control Conference ECC'99*, paper F1032-5(CD-ROM), Karlsruhe, 1999.
5. M.S. Bazaraa, J. Sherali, and K. Shetty. *Nonlinear Programming: Theory and Algorithms*. J.Wiley & Sons, New York, 1993.
6. V. M. Becerra, P.D. Roberts, and G. W. Griffiths. Novel developments in process optimisation using predictive control. *J. Process Control*, 8(2):117–138, 1998.
7. A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear-quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.
8. B.W. Bequette. Nonlinear control of chemical processes: a review. *Ind. Engng. Chem. Res.*, 30:1391–1413, 1991.
9. D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, 1995.
10. Y. Blanco, W. Perruquetti, and P. Borne. Non quadratic stability of nonlinear systems in the Takagi-Sugeno form. In *Proc. 6th European Control Conference*, pages 3917–3922, Porto, Portugal, 2001.
11. T. L. Blevins, G. K. McMillan, W. K. Wojsznis, and M. W. Brown. *Advanced Control Unleashed*. ISA, 2003.
12. M. Brdyś. Hierarchical optimizing control of steady-state large-scale systems under model-reality differences of mixed type – a mutually interacting approach. In *3rd IFAC Symp. Large Scale Systems Th. and Appl.*, pages 49–57, Warszawa, 1984.
13. M. Brdyś, J.E. Ellis, and P.D. Roberts. Augmented integrated system optimization and parameter estimation technique: derivation, optimality and convergence. *IEE Proc. Pt. D*, 34(3):201–209, 1987.

14. M. Brdyś and P.D. Roberts. Convergence and optimality of modified two-step algorithm for integrated system optimization and parameter estimation. *Int. J. Systems Sci.*, 18(7):1305–1322, 1987.
15. M. Brdyś and P. Tatjewski. An algorithm for steady-state optimising dual control of uncertain plants. In *Proc. 1st IFAC Works. New Trends in Design of Control Systems*, pages 249–254, Smolenice, Slovakia, 1994.
16. M.A. Brdys and P. Tatjewski. *Iterative Algorithms for Multilayer Optimizing Control*. Imperial College Press/World Scientific, London/Singapore, 2005.
17. E.F. Camacho and C. Bordons. *Model Predictive Control in the Process Industry*. Springer, Berlin, 1995.
18. E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer, London, 1999.
19. M. Cannon, B. Kouvaritakis, Y.I. Lee, and A.C. Brooms. Efficient nonlinear model-based predictive control. *Int. J. Control*, 74(4):361–372, 2001.
20. S.G. Cao, N.W. Rees, and G. Feng. Stability analysis of fuzzy control systems. *IEEE Trans. Systems, Man and Cybernetics – Part B: Cybernetics*, 26(1):201–204, 1996.
21. M. Chadli, D. Maquin, and J. Ragot. Relaxed stability conditions for Takagi-Sugeno fuzzy systems. In *Proc. IEEE Conf. Systems, Man and Cybernetics*, pages 3514–3519, 2000.
22. M. Chadli, D. Maquin, and J. Ragot. On the stability analysis of multiple model systems. In *Proc. 6th European Control Conference ECC'01*, pages 1894–1899, Porto, 2001.
23. J.-S. Chai, S. Tan, Q. Chen, and C.-C. Hang. A general fuzzy control scheme for nonlinear processes with stability analysis. *Fuzzy Sets and Systems*, 100:179–195, 1998.
24. C.C. Chen and L. Shaw. On receding horizon feedback control. *Automatica*, 18(3):349–352, 1982.
25. H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217, 1998.
26. D.W. Clarke and C. Mohtadi. Properties of generalised predictive control. *Automatica*, 25:859–875, 1989.
27. D.W. Clarke, C. Mohtadi, and P.S. Tuffs. Generalised predictive control – Parts I and II. *Automatica*, 23:137–160, 1987.
28. D.W. Clarke and R. Scattolini. Constrained receding-horizon predictive control. *IEE Proc. Pt. D*, 138(4):347–354, 1991.
29. C.R. Cutler and B.L. Ramaker. Dynamic Matrix Control – a computer control algorithm. In *Proc. Joint Automatic Control Conference*, San Francisco, 1980.
30. F. Declercq and R. de Keyser. Suboptimal nonlinear predictive controllers. *Int. J. Appl. Math. and Comp. Sci.*, 9(1):129–148, 1999.
31. P. Domański. *Zastosowanie metod jakościowych do modelowania i projektowania układów regulacji (Application of qualitative methods for modeling and design of feedback control systems,* in Polish*)*. PhD thesis, Warsaw University of Technology, Faculty of Electronics and Information Technology, Warszawa, 1996.
32. P. Domański, M. A. Brdyś, and P. Tatjewski. Design and stability of fuzzy-logic multi-regional output controllers. *Int. J. Appl. Math. and Comp. Sci.*, 9(4):883–897, 1999.

33. P. Domański, M.A. Brdyś, and P. Tatjewski. Fuzzy logic multi-regional controllers – design and stability analysis. In *Proc. 4th European Control Conference ECC'97*, paper TU-E G5 (CD-ROM), Brussels, 1997.
34. F.J. Doyle III, B.A. Ogunnaike, and R.K. Pearson. Nonlinear model-based control using second-order Volterra models. *Automatica*, 31(5):697–714, 1995.
35. J. Duda, M.A. Brdyś, P. Tatjewski, and W. Byrski. Multilayer decomposition for optimizing control of technological processes. In *Proc. IFAC/IFORS/IMACS Symp. Large Scale Systems Th. and Appl. LSS'95*, pages 111–116, London, 1995.
36. H.H. Eder. MBPC benefits and key success factors. In *Proc. 5th European Control Conference ECC'99*, paper F1033-6 (CD-ROM), Karlsruhe, 1999.
37. W. Findeisen. *Wielopoziomowe Układy Sterowania (Multilayer Control Systems,* in Polish*)*. PWN, Warszawa, 1974.
38. W. Findeisen. *Technika Regulacji Automatycznej (Feedback Control Techniques,* in Polish*)*. PWN, Warszawa, 1978.
39. W. Findeisen. *Struktury Sterowania dla Złożonych Procesów (Control Structures for Complex Processes,* in Polish*)*. Warsaw University of Technology Press, Warszawa, 1997.
40. W. Findeisen, F. N. Bailey, M. Brdyś, K. Malinowski, P. Tatjewski, and A. Woźniak. *Control and Coordination in Hierarchical Systems*. J. Wiley & Sons, Chichester - New York - Brisbane - Toronto, 1980.
41. W. Findeisen, M. Brdyś, K. Malinowski, P. Tatjewski, and A.Woźniak. On-line hierarchical control for steady-state systems. *IEEE Trans. Automatic Control*, 23(2):189–208, 1978.
42. M. Fischer, O. Nelles, and R. Isermann. Adaptive predictive control of a heat exchanger based on a fuzzy model. *Control Eng. Practice*, 6:259–269, 1998.
43. R.R. Fletcher. *Practical Methods of Optimization*. J. Wiley & Sons, Chichester - New York - Brisbane - Toronto, 1987.
44. G. Franklin, D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Wesley, Reading, Massachusetts, 1994.
45. G.F. Franklin, J.D. Powell, and M.L. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, Reading, Massachusetts, 1994.
46. R.G.E. Franks. *Mathematical Modelling in Chemical Engineering*. J. Wiley, 1972.
47. C.E. Garcia. Quadratic/dynamic martix control of nonlinear processes: an application to a batch reaction process. In *Proc. AIChE Annual Meeting*, San Francisco, 1984.
48. C.E. Garcia and A.M. Morshedi. Quadratic programming solution of dynamic matrix control (QDMC). *Chemical Eng. Communications*, 46:73–87, 1986.
49. C.E. Garcia, D.M. Prett, and M. Morari. Model predictive control: theory and practice – a survey. *Automatica*, 25:335–348, 1989.
50. G. Gattu and E. Zafiriou. Nonlinear quadratic dynamic matrix control with state estimation. *Ind. Eng. Chem. Res.*, 31:1096–1104, 1992.
51. G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1989.
52. G. C. Goodwin, S. F. Graebe, and M. E. Salgado. *Control System Design*. Prentice Hall, Upper Saddle River, 2001.
53. H. Haimovich, M.M. Seron, G.C.Goodwin, and J.C. Agüero. A neural approximation to the explicit solution of constrained linear MPC. In *Proc. 7th European Control Conference ECC'03*, paper 183 (CD ROM), Cambridge, 2003.

54. S. Haykin. *Neural Networks - A Comprehensive Foundation.* Prentice Hall, Englewood Cliffs, NY, 1999.
55. M.A. Henson. Nonlinear model predictive control: current status and future directions. *Computers and Chemical Eng.*, 23:187–202, 1998.
56. P. Hoekstra, T.J.J. van den Boom, and M.A. Botto. Design of an analytic constrained predictive controller using neural networks. In *Proc. 6th European Control Conference ECC'01*, paper 3895(CD-ROM), Porto, 2001.
57. A. Isidori. *Nonlinear Control Systems.* Springer, Berlin, 1995.
58. J.-S. R. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing.* Prentice Hall, Upper Saddle River, 1997.
59. J.-S.R. Jang and C.-T. Sun. Neuro-fuzzy modeling and control. *Proc. of IEEE*, 83(3):378–406, 1995.
60. M. Johansson, A. Rantzer, and K.-E. Arzen. Piecewise quadratic stability of fuzzy systems. *IEEE Trans. Fuzzy Systems*, 7(6):713–722, 1999.
61. R.E. Kalman and J.E. Bertram. Control system analysis and design via the second method of Lyapunov. I Continuous-time systems. II Discrete-time systems. *J. Basic Eng. Trans. ASME*, 82(2):371–400, 1960.
62. D. E. Kassmann, T.A. Badgwell, and R.B. Hawkins. Robust steady-state target calculation for model predictive control. *AIChE Journal*, 46:1007–1024, 2000.
63. S.S. Keerthi and E.G. Gilbert. Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving horizons approximations. *J. Optimiz. Th. and Appl.*, 57:265–293, 1988.
64. H.A. Khalil. *Nonlinear Systems.* Prentice Hall, Upper Saddle River, 1996.
65. J. Korbicz, J. M. Kościelny, Z. Kowalczuk, and W. Cholewa, editors. *Fault Diagnosis. Models, Artificial Intelligence, Applications.* Springer, London, 2004.
66. W.H. Kwon and A.E. Pearson. A modified quadratic cost problem and feedback stabilization of a linear system. *IEEE Trans. Automatic Control*, 22(5):838–842, 1977.
67. M. Ławryńczuk. *Nieliniowe algorytmy regulacji predykcyjnej z neuronowymi modelami procesów (Nonlinear predictive control algorithms with neural models of processes,* in Polish*).* PhD thesis, Warsaw University of Technology, Faculty of Electronics and Information Technology, Warszawa, 2003.
68. M. Ławryńczuk, P. Marusak, and P. Tatjewski. Oprogramowanie do projektowania zaawansowanych układów regulacji (Software for advanced feedback control systems design, in Polish). In *Proc. XIV Polish Control Conference*, pages 483–488, Zielona Góra, Poland, 2002.
69. M. Ławryńczuk, P. Marusak, and P. Tatjewski. Optimizing predictive range control for a distillation process. In *Proc. 11th IEEE Conf. Methods and Models in Automation and Robotics MMAR'05*, pages 379–384, Miedzyzdroje, Poland, 2005.
70. M. Ławryńczuk, P. Marusak, and P. Tatjewski. Integrating predictive control with steady-state optimization. In *Proc. 12th IEEE Conf. Methods and Models in Automation and Robotics MMAR'06*, Miedzyzdroje, Poland, 2006.
71. M. Ławryńczuk and P. Tatjewski. A multivariable neural predictive control algorithm. In *Proc. IFAC Works. Advanced Fuzzy-Neural Control AFNC'01*, pages 191–196, Valencia, 2001.
72. M. Ławryńczuk and P. Tatjewski. A nonlinear predictive control algorithm for processes modelled by means of neural networks. In *Proc. 7th IEEE Conf. on*

*Methods and Models in Automation and Robotics MMAR'01*, volume 1, pages 489–494, Miedzyzdroje, 2001.
73. M. Ławryńczuk and P. Tatjewski. A computationally efficient nonlinear predictive control algorithm based on neural models. In *Proc. 8th IEEE Conf. on Methods and Models in Automation and Robotics MMAR'02*, pages 781–786, Szczecin, 2002.
74. M. Ławryńczuk and P. Tatjewski. An infinite horizon predictive control algorithm based on multivariable input-output models. *Int. J. Appl. Math. Comp. Sci.*, 14(2):167–180, 2004.
75. M. Ławryńczuk and P. Tatjewski. A stable dual-mode nonlinear predictive control algorithm based on on-line linearization and quadratic programming. In *Proc. 10th IEEE Conf. on Methods and Models in Automation and Robotics MMAR'04*, volume 1, pages 503–510, Miedzyzdroje, 2004.
76. E.B. Lee and L. Markus. *Foundations of optimal control theory.* J. Wiley, New York, 1967.
77. J.H. Lee and N.L. Ricker. Extended Kalman filter based nonlinear model predictive control. *Ind. Eng. Chem. Res.*, 33:1530–1541, 1994.
78. I. Lefkowitz. Multilevel approach applied to control system design. *J. Basic Eng. Trans. ASME, ser. B*, 2:392–398, 1966.
79. W.C. Li and T. Biegler. Multistep, Newton-type control strategies for constrained, nonlinear processes. *Chem. Eng. Res. Des.*, 67:562–577, 1989.
80. G.P. Liu, V. Kadirkamanathan, and S.A. Billings. Predictive control for nonlinear systems using neural networks. *Int. J. Control*, 71(6):1119–1132, 1998.
81. W.L. Luyben. *Process Modeling, Simulation, and Control for Chemical Engineers.* McGraw-Hill, New York, 1973.
82. J.M. Maciejowski. *Predictive Control.* Prentice Hall, Harlow, England, 2002.
83. L. Magni, G. De Nicolao, L. Magnani, and R. Scattolini. A stabilizing model-based predictive control algorithm for nonlinear systems. *Automatica*, 37:1351–1362, 2001.
84. B.R. Maner, F.J. DoyleIII, B.A. Ogunnaike, and R.K. Pearson. Nonlinear model predictive control of a simulated multivariable polymerization reactor using second-order Volterra models. *Automatica*, 32(9):1285–1301, 1996.
85. T. E. Marlin. *Process Control.* McGraw-Hill, New York, 1995.
86. P. Marquis and J.P. Broustail. SMOC, a bridge between state space and model predictive controllers: Application to the automation of a hydrotreating unit. In T.J. McAvoy, Y. Arkun, and E. Zafiriou, editors, *Proc. 1988 IFAC Works. Model Based Process Control*, pages 37–43. Pergamon Press, Oxford, 1988.
87. P. Marusak. *Regulacja predykcyjna obiektów nieliniowych z wykorzystaniem techniki macierzy dynamicznej (Predictive control of nonlinear processes using dynamic matrix techniques,* in Polish*).* PhD thesis, Warsaw University of Technology, Faculty of Electronics and Information Technology, Warszawa, 2002.
88. P. Marusak, J. Pułaczewski, and P. Tatjewski. Algorytmy DMC z uwzględnieniem ograniczeń sterowania (DMC algorithms with constraints on control inputs, in Polish). In *Proc. XIII Polish Control Conference*, volume 1, pages 113–118, Opole, 1999.
89. P. Marusak and P. Tatjewski. Fuzzy Dynamic Matrix Control algorithms for nonlinear plants. In *Proc. 6th Int. Conf. Methods and Models in Automation and Robotics MMAR'00*, pages 749–754, Miedzyzdroje, Poland, 2000.

90. P. Marusak and P. Tatjewski. Output constraints in fuzzy DMC algorithms with parametric uncertainty in process models. In *Proc. 7th IFAC Conf. Methods and Models in Automation and Robotics MMAR'01*, volume 1, pages 517–522, Miedzyzdroje, 2001.
91. P. Marusak and P. Tatjewski. Stability analysis of nonlinear control systems with fuzzy DMC controllers. In *Proc. IFAC Works. Advanced Fuzzy and Neural Control AFNC'01*, pages 21–26, Valencia, 2001.
92. P. Marusak and P. Tatjewski. Stability analysis of nonlinear control systems with unconstrained fuzzy predictive controllers. *Archives of Control Sciences*, 12:267–288, 2002.
93. P. Marusak and P. Tatjewski. Stable, effective fuzzy DMC algorithms with online quadratic optimization. In *Proc. American Control Conference ACC2003*, pages 3513–3518, Denver,Colorado, 2003.
94. D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: stability and optimality. *Automatica*, 36:789–814, 2000.
95. D. Megias, J. Serrano, and M.Y. El Ghoumari. Extended linearized predictive control: practical control algorithms for non-linear systems. In *Proc. 5th European Control Conference ECC'99*, paper F883 (CD-ROM), Karlsruhe, 1999.
96. M. D. Mesarović, D. Macko, and Y. Takahara. *Theory of Hierarchical, Multilevel Systems.* Academic Press, New York, 1970.
97. H. Michalska and D.Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Trans. Automatic Control*, 38:1623–1633, 1993.
98. M. Morari and J.H. Lee. Model predictive control: past, present and future. *Computers and Chemical Eng.*, 23:667–682, 1999.
99. M. Morari and N.L. Ricker. *Model Predictive Control Toolbox, User's Guide ver. 1.* The MathWorks Inc., Natick, MA, 1998.
100. K.R. Muske and J.B. Rawlings. Model predictive control with linear models. *AICHE Journal*, 39(2):262–287, 1993.
101. R.K. Mutha, W.R. Cluett, and A. Penlidis. Nonlinear model-based predictive control of control nonaffine systems. *Automatica*, 33(5):907–913, 1997.
102. G. De Nicolao, L. Magni, and R. Scattolini. Stabilizing receding-horizon control of nonlinear time-varying systems. *IEEE Trans. Automatic Control*, 43(7):1030–1036, 1998.
103. A. Niederliński, J. Mościński, and Z. Ogonowski. *Regulacja Adaptacyjna (Adaptive Control,* in Polish*).* PWN, Warszawa, 1995.
104. K. Nowosad. *Strukturalne Cechy Regulacji Predykcyjnej (Structural Properties of Predictive Control,* in Polish*).* Warsaw University of Technology Press, Warszawa, 1995.
105. A. O'Dwyer. *PI and PID Controller Tuning Rules.* Imperial College Press/World Scientific, London/Singapore, 2003.
106. K. Ogata. *Modern Control Engineering.* Prentice Hall, Englewood Cliffs, NJ, 1997.
107. T. Parisini and S. Sacone. Stable hybrid control based on discrete-event automata and receding-horizon neural regulators. *Automatica*, 37(8):1279–1292, 2001.
108. T. Parisini, M. Sanguinetti, and R. Zoppoli. Nonlinear stabilization by receding-horizon neural regulators. *Int. J. Control*, 70(3):341–362, 1998.
109. T. Parisini and R. Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10):1443–1451, 1995.

110. S. Piche, B. Sayyar-Rodsari, D. Johnson, and M. Gerules. Nonlinear model predictive control using neural networks. *IEEE Control Systems Magazine*, 20(3):56–62, 2000.
111. A. Piegat. *Modelowanie i Sterowanie Rozmyte (Fuzzy Modeling and Control,* in Polish*)*. EXIT Academic Publ., Warszawa, 1999.
112. A.N. Płanowski, W.M. Ramm, and S.Z. Kagan. *Procesy i Aparaty w Technologii Chemicznej, (Processes and Apparatuses in Chemical Technology,* in Polish*)*. PWN, Warszawa, 1974.
113. D.M. Prett and R.D. Gillette. Optimization and constrained multivariable control of catalytic cracking unit. In *Proc. Joint Automatic Control Conference*, San Francisco, 1980.
114. J. Pułaczewski, Z. Komor, W. Szkolnikowski, and T. Mucai. Nowy regulator rodziny EFTRONIK - EFTRONIK XF (New controller of EFTRONIK family - EFTRONIK XF, in Polish). *Pomiary, Automatyka, Kontrola*, 1998(3):63–67, 1998.
115. S.J. Qin and T.A. Badgwell. A survey of industrial model predictive control technology. *Control Eng. Practice*, 11:733–764, 2003.
116. A. Rantzer and A. Johansson. Piecewise linear-quadratic optimal control. *IEEE Trans. Automatic Control*, 45(4):629–637, 2000.
117. J.B. Rawlings and K.R. Muske. The stability od constrained receding horizon control. *IEEE Trans. Automatic Control*, 38:1512–1516, 1993.
118. M. Nørgaard, O. Ravn, N.K. Poulsen, and L.K. Hansen. *Neural Networks for Modeling and Control of Dynamic Systems*. Springer, London, 2000.
119. J. Richalet, A. Rault, J.L. Testud, and J. Papon. Algorithmic control of industrial processes. In *Proc. 4th IFAC Symp. Identification and System Parameter Estimation*, pages 1119–1167, 1976.
120. J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14:413–428, 1978.
121. P.D. Roberts. An algorithm for steady-state optimization and parameter estimation. *Int. J. Systems Sci.*, 10:719–734, 1979.
122. P.D. Roberts and T.W.C. Williams. On an algorithm for combined system optimization and parameter estimation. *Automatica*, 17(4):455–472, 1981.
123. J.A. Rossiter. *Model-Based Predictive Control*. CRC Press, Boca Raton - London - New York - Washington,D.C., 2003.
124. J.A. Rossiter, M.J. Rice, and B. Kouvaritakis. A numerically robust state-space approach to stable predictive control strategies. *Automatica*, 34:65–73, 1998.
125. P.O.M. Scokaert. Infinite horizon generalized predictive control. *Int. J. Control*, 66(5):161–175, 1997.
126. P.O.M. Scokaert and D.W. Clarke. Stabilising properties of constrained predictive control. *IEE Proc. – Control Theory Appl.*, 141(5):295–304, 1994.
127. P.O.M. Scokaert, D.Q. Mayne, and J.B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Trans. Automatic Control*, 44(3):648–654, 1999.
128. M. Setnes and J.M. Sousa. Fuzzy rule-based optimization in nonlinear predictive control. In *Proc. 5th European Control Conference ECC'99*, paper F907(CD-ROM), Karlsruhe, 1999.
129. A. Stachurski and A.P. Wierzbicki. *Podstawy Optymalizacji (Foundations of Optimization,* in Polish*)*. Warsaw University of Technology Press, Warszawa, 1999.

130. W. Tadej. *Analiza własności dualnego algorytmu optymalizacji punktów pracy procesów nieliniowych (Analysis of dual algorithm for set-point optimization of nonlinear processes,* in Polish*)*. PhD thesis, Warsaw University of Technology, Faculty of Electronics and Information Technology, Warszawa, 2001.
131. T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modeling and control. *Trans. Systems, Man and Cybernetics*, 15(1), 1985.
132. K. Tanaka, T. Ikeda, and H.O. Wang. Fuzzy regulators and fuzzy observers: relaxed stability conditions and LMI-based designs. *IEEE Trans. Fuzzy Systems*, 6(2):250–265, 1998.
133. K. Tanaka and M. Sugeno. Stability analysis and design of fuzzy control systems. *Fuzzy Sets and Systems*, 45:135–156, 1992.
134. P. Tatjewski. *Sterowanie Zaawansowane Procesów Przemysłowych, Struktury i Algorytmy (Advanced Control of Industrial Processes, Structures and Algorithms,* in Polish*)*. EXIT Academic Publ., Warszawa, 2002.
135. P. Tatjewski, M.A. Brdyś, and J. Duda. Optimising control of uncertain plants with constrained, feedback controlled outputs. *Int. J. Control*, 74(15):1510–1526, 2001.
136. P. Tatjewski and M. Ławryńczuk. Soft computing in model-based predictive control. *Int. J. Appl. Math. Comp. Sci.*, 16(1):7–26, 2006.
137. P. Tatjewski, M. Ławryńczuk, and P. Marusak. Wybrane algorytmy regulacji predykcyjnej procesów nieliniowych (Selected algorithms of nonlinear predictive control, in Polish). In *Proc. XIV Polish Control Conference*, pages 145–152, Zielona Góra, Poland, 2002.
138. P. Tatjewski, M. Ławryńczuk, and P. Marusak. Linking nonlinear steady-state and target set-point optimization for model predictive control. In *Proc. Int. Conf. Control 2006*, Glasgow, UK, 2006.
139. P. Tatjewski and P.D. Roberts. Newton-like algorithm for integrated system optimization and parameter estimation technique. *Int. J. Control*, 46(4):1155–1170, 1987.
140. M.C. Teixeira, E. Assunçao, and H.C. Pietrobom. On relaxed LMI-based designs for fuzzy regulators and fuzzy observers. In *Proc. 6th European Control Conference ECC'01*, pages 120–125, Porto, Portugal, 2001.
141. M.C.M. Teixeira and S.H. Żak. Stabilizing controller design for uncertain nonlinear systems using fuzzy models. *IEEE Trans. Fuzzy Systems*, 7(2):133–142, 1999.
142. W.K. Wojsznis T.L. Blevins, G.K. McMillan and M.W. Brown. *Advanced Control Unleashed*. The ISA Society, Research Triangle Park, NC, 2003.
143. P. Tondel, T.A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. *Automatica*, 39:489–497, 2003.
144. P. Tondel, T.A. Johansen, and A. Bemporad. Evaluation of piecewise affine control via binary search tree. *Automatica*, 39:945–950, 2003.
145. M. Tvrzska de Gouvea and D. Odloak. One-layer real time optimization of LPG production in the FCC unit: procedure, advantages and disadvantages. *Computers and Chemical Eng.*, 22:191–198, 1998.
146. J. Vada, O. Slupphaug, T.A. Johansen, and B.A. Foss. Linear MPC with optimal prioritized infeasibility handling: application, computational issues and stability. *Automatica*, 37:1835–1843, 2001.

147. P.J. van der Veen, R. Babuska, and H.B. Verbruggen. Comparison of nonlinear predictive control methods for a waste-water treatment benchmark. In *Proc. 5th European Control Conference ECC'99*, paper F1060-1 (CD-ROM), Karlsruhe, 1999.
148. M. Vidyasagar. *Nonlinear Systems Analysis*. Prentice Hall, Englewood Cliffs, N.J., 1993.
149. J.P. Vila and V. Wagner. Predictive neuro-control of uncertain systems: design and use of a neuro-optimizer. *Automatica*, 39(5):767–777, 2001.
150. H.O. Wang, K. Tanaka, and M.F. Griffin. An approach to fuzzy control of nonlinear systems: stability and design issues. *IEEE Trans. Fuzzy Systems*, 4(1):14–23, 1996.
151. L. Wang and G. Feng. Piecewise $H_\infty$ controller design of discrete time fuzzy systems. *IEEE Trans. Systems, Man, and Cybernetics – Part B: Cybernetics*, 34(1):682–686, 2004.
152. L.X. Wang and F. Wan. Structured neural networks for constrained model predictive control. *Automatica*, 37(8):1235–1243, 2001.
153. H. P. Williams. *Model Building in Mathematical Programming*. J. Wiley, Chichester-New York, 1995.
154. M. Wonham. *Linear Multivariable Systems*. Springer, Berlin, 1974.
155. J. Yen and R. Langari. *Fuzzy Logic - Intelligence, Control and Information*. Prentice Hall, Upper Saddle River, New Jersey, 1999.
156. L.A. Zadeh. Fuzzy sets. *Information and Control*, 8, 1965.
157. E. Zafiriou. Robust model predictive control of processes with hard constraints. *Computers and Chemical Eng.*, 14(4/5):359–371, 1990.
158. A. Zanin, M. Tvrzska de Gouvea, and D. Odloak. Industrial implementation of a real-time optimization strategy for maximizing production of LPG in a FCC unit. *Computers and Chemical Eng.*, 24:525–531, 2000.
159. A. Zanin, M. Tvrzska de Gouvea, and D. Odloak. Integrating real-time optimization into model predictive controller of the FCC system. *Control Eng. Practice*, 10:819–831, 2002.
160. H. Zhang and P.D. Roberts. On-line steady-state optimization of nonlinear constrained processes with slow dynamics. *Trans. Inst. MC*, 12(5):251–261, 1990.
161. J. Zhao, V. Wertz, and R. Gorez. Dynamic fuzzy state-feedback controller and its limitations. In *Proc. 13th IFAC Trennial World Congress*, pages 121–126, San Francisco, 1996.

# Index