

Liuping Wang



Advances in
Industrial Control

Model Predictive Control System Design and Implementation Using MATLAB®



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

Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques

Silvio Simani, Cesare Fantuzzi and Ron J. Patton

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

Ship Motion Control

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 C. 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_∞ Constrained Feedback Control

Murad Abu-Khalaf, Jie Huang and Frank L. Lewis

Practical Grey-box Process Identification

Torsten Bohlin

Control of Traffic Systems in Buildings

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

(continued after Index)

Liuping Wang

Model Predictive Control System Design and Implementation Using MATLAB[®]



Springer

Liuping Wang, PhD
School of Electrical and Computer Engineering
RMIT University
Melbourne, VIC 3000
Australia

ISBN 978-1-84882-330-3

e-ISBN 978-1-84882-331-0

DOI 10.1007/978-1-84882-331-0

Advances in Industrial Control ISSN 1430-9491

A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2008940691

© 2009 Springer-Verlag London Limited

MATLAB® and Simulink® are registered trademarks of The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, USA. <http://www.mathworks.com>

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms of licences issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers.

The use of registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant laws and regulations and therefore free for general use.

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

Cover design: eStudio Calamar S.L., Girona, Spain

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Advances in Industrial Control

Series Editors

Professor Michael J. Grimble, Professor of Industrial Systems and Director

Professor Michael A. Johnson, Professor (Emeritus) of Control Systems and Deputy Director

Industrial Control Centre

Department of Electronic and Electrical Engineering

University of Strathclyde

Graham Hills Building

50 George Street

Glasgow G1 1QE

United Kingdom

Series Advisory Board

Professor E.F. Camacho

Escuela Superior de Ingenieros

Universidad de Sevilla

Camino de los Descubrimientos s/n

41092 Sevilla

Spain

Professor S. Engell

Lehrstuhl für Anlagensteuerungstechnik

Fachbereich Chemietechnik

Universität Dortmund

44221 Dortmund

Germany

Professor G. Goodwin

Department of Electrical and Computer Engineering

The University of Newcastle

Callaghan

NSW 2308

Australia

Professor T.J. Harris

Department of Chemical Engineering

Queen's University

Kingston, Ontario

K7L 3N6

Canada

Professor T.H. Lee

Department of Electrical and Computer Engineering

National University of Singapore

4 Engineering Drive 3

Singapore 117576

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
221 00 Lund
Sweden

Professor A. Ray
Department of Mechanical Engineering
Pennsylvania State University
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 and Computer Engineering
National University of Singapore
4 Engineering Drive 3
Singapore 117576

Professor I. Yamamoto
Department of Mechanical Systems and Environmental Engineering
The University of Kitakyushu
Faculty of Environmental Engineering
1-1, Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka, 808-0135
Japan

In memory of my parents

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.

Today's control software and technology offers the potential to implement more advanced control algorithms but often the preferred strategy of many industrial engineers is to design a robust and transparent process control structure that uses simple controllers. This is one reason why the PID controller remains industry's most widely implemented controller despite the extensive developments of control theory; however, this approach of structured control can create limitations on good process performance. One such limitation is the possible lack of a coordinator within the hierarchy that systematically achieves performance objectives. Another is the omission of a facility to accommodate and handle process operational constraints easily. The method of model predictive control (MPC) can be used in different levels of the process control structure and is also able to handle a wide variety of process control constraints systematically. These are two of the reasons why MPC is often cited as one of the more popular advanced techniques for industrial process applications.

Surprisingly, MPC and the associated receding horizon control principle have a history of development and applications going back to the late 1960s; Jacques Richalet developed his predictive functional control technique for industrial application from that time onward. Work on using the receding horizon control concept with state-space models can be identified in the literature of the 1970s, and the 1980s saw the emergence first of dynamic matrix control and then, towards the end of the decade, of the influential generalised predictive control technique.

This field continues to develop and the *Advances in Industrial Control* monograph series has several volumes on the subject. These include *Applied*

Predictive Control by S. Huang, K.K. Tan and T.H. Lee (ISBN 978-1-85233-338-6, 2002), *Fuzzy Logic, Identification and Predictive Control* by J.J. Espinosa, J.P.L. Vandewalle and V. Wertz (ISBN 978-1-85233-828-2, 2005) and *Advanced Control of Industrial Processes* by P. Tatjewski (ISBN 978-1-84628-634-6, 2007). In our related series, *Advanced Textbooks in Control and Signal Processing*, we have published *Model Predictive Control* by E.F. Camacho and C. Bordons (2nd edition, ISBN 978-1-85233-694-3, 2004), and *Receding Horizon Control* (ISBN 978-1-84628-024-5, 2005) by W.H. Kwon and S. Han.

To the above group of books we are now able to add this monograph, *Model Predictive Control System Design and Implementation Using MATLAB®*, by Liuping Wang. Professor Wang aims to provide both the industrial and the academic reader with a direct but graded route into understanding MPC as used in the solution of industrial control problems. The interleaved exposition, and MATLAB® tutorials, allow the reader to work through a structured introduction to the design and implementation of MPC and use some related tools to condition, tune and test the control design solutions.

Some features of MPC that makes it worthy of study as an industrial control technique include:

- the technique uses simple concepts;
- the controller tuning can be packaged for ease of use;
- the technique can be used in either supervisory or primary control modes; and
- constraint handling is naturally accommodated by the method, and can be packaged for automated systematic constraint setup.

Professor Wang's book illustrates these issues and uses a small set of theoretical tools to great effect; these tools include the exponential weighting of signals, weights to achieve a prescribed degree of stability, re-parameterisation of the feedback using orthogonality principles, Laguerre and Kautz basis functions, and quadratic programming. The book is structured so that discrete methods are considered first, and these are followed by continuous-time system techniques. Over the course of the book, the MATLAB® interludes result in readers constructing their own libraries of MPC routines that can be used in other control problems. Towards the end of the book, Professor Wang demonstrates the use of the MPC algorithms in some application studies. These range from a motor control application to the control of a food extruder process and these studies illustrate both the software and hardware aspects of the solutions.

The book's "hands-on" approach is expected to appeal to a wide readership ranging from the industrial control engineer to the postgraduate student in the process and control disciplines. Both will undoubtedly find the MATLAB® demonstrations of the control concepts an invaluable tutorial route to understanding MPC in practice.

Industrial Control Centre
Glasgow
Scotland, UK
2008

M.J. Grimble
M.A. Johnson

Preface

About this Book

Model predictive control (MPC) has a long history in the field of control engineering. It is one of the few areas that has received on-going interest from researchers in both the industrial and academic communities. Four major aspects of model predictive control make the design methodology attractive to both practitioners and academics. The first aspect is the design formulation, which uses a completely multivariable system framework where the performance parameters of the multivariable control system are related to the engineering aspects of the system; hence, they can be understood and ‘tuned’ by engineers. The second aspect is the ability of the method to handle both ‘soft’ constraints and hard constraints in a multivariable control framework. This is particularly attractive to industry where tight profit margins and limits on the process operation are inevitably present. The third aspect is the ability to perform on-line process optimization. The fourth aspect is the simplicity of the design framework in handling all these complex issues.

This book gives an introduction to model predictive control, and recent developments in design and implementation. Beginning with an overview of the field, the book will systematically cover topics in receding horizon control, MPC design formulations, constrained control, Laguerre-function-based predictive control, predictive control using exponential data weighting, reformulation of classical predictive control, tuning of predictive control, as well as simulation and implementation using MATLAB® and SIMULINK® as a platform. Both continuous-time and discrete-time model predictive control is presented in a similar framework.

Development of this Book

There are several aspects of the developments that may be of interest to the reader.

Theory

This book was originally planned as a research monograph to cover the methodologies of predictive control using orthonormal basis functions. Design of predictive control using time-domain functions is not new. It was proposed by Richalet in the 1970s and has been successfully used in various control engineering applications (see Richalet *et al.*, 1978, Richalet, 1987, 1993, 2000). However, the design of predictive control that uses orthonormal functions is new, particularly the approaches that use the sets of exponential orthonormal functions such as Laguerre functions and Kautz functions. With my background in system identification, I naturally saw the link between system identification and predictive control from the perspective of modelling of the control trajectory. Once the predictive control problem is formulated as that of modelling the future control trajectory, both continuous-time and discrete-time predictive control is solved in the same framework. Both have direct links to the classical linear quadratic regulator (LQR) in continuous time and discrete time when using a sufficiently long prediction horizon. The key difference between predictive control and LQR is that the predictive control solves the optimization problem using a moving time horizon window whilst LQR solves the same problem within a fixed window. The advantages of using a moving time horizon window include the ability to perform real-time optimization with hard constraints on plant variables.

As we know, the majority of industrial control systems have integral action (*e.g.*, PID controllers). This integral functionality has also been embedded in the classical predictive control systems such as generalized predictive control (GPC) and dynamic matrix control (DMC) (see Clarke *et al.*, 1987, Cutler and Ramaker, 1979). In this book, in order to have an integral function in the MPC algorithms, the design models are embedded with integrators. The formulation of the design models was inspired by the state-space approach given in Ricker's paper (Ricker, 1991). By doing so, similar to GPC and DMC, the optimized control trajectory is either the increment of the control signal (discrete-time case) or the derivative of the control signal (continuous-time case). The added benefit is the simplified implementation procedure, where less steady-state plant information is required.

One of the well-known problems in classical predictive control is its numerical problem when the prediction horizon is large (for example, Rossiter *et al.*, 1998). This was inevitable because the model used for design contained an integrating mode. To overcome this problem, the design model needs to be asymptotically stable. Using the classical results in LQR by Anderson and Moore (see Anderson and Moore, 1971), a simple transformation of the design model from unstable to stable was made by choosing a cost that contains an exponential factor, hence overcoming the numerical ill-conditioning problem (see Wang 2001c). In order to guarantee closed-loop stability of the predictive control system, the original weight matrices in the cost function are adjusted when the exponential weight is used. An important by-product

of this transformation is the creation of a prescribed degree of stability in the MPC algorithms, both continuous-time and discrete-time. This prescribed degree of stability in conjunction with the use of orthonormal functions made the tuning of the predictive control system a relatively easy task.

The idea of creating a prescribed degree of stability has been previously used in receding horizon control (see Kwon and Han, 2005, Yoon and Clarke, 1993), and the key in the previous approaches relies on the use of an exponentially increasing weight in the cost function, along with the solution obtained analytically from a Riccati approach. The distinguishing point for the approaches in this book is that instead of increasing, an exponentially decreasing weight is used in the cost function to resolve the numerical problem, followed by adjustment of the constant weight matrices to recover and create more stability margins if required. Use of an exponentially decreasing weight is counter-intuitive from a closed-loop stability point of view, however, it makes sense when the numerical issue is of concern.

Practice

Predictive control using orthonormal basis functions has been successfully used in numerous applications, mainly by my previous undergraduate and postgraduate students. The majority of the applications were based on MATLAB real-time Workshop and SIMULINK xPC target. The MATLAB programs I wrote over the years are often directly translated into MATLAB C or SIMULINK programs for xPC target, then the applications followed through. The MATLAB programs are useful for practitioners as a start point, adopting them later for their own applications. The MATLAB programs are explained in a tutorial manner, step by step, so that the reader can understand how the algorithms are developed.

Presentation

The first version of this book had a different structure from the current version. The book had begun with continuous-time predictive control using orthonormal basis functions. The pre-conference predictive control workshop in the American Control Conference (ACC) 2007 made me change my mind about the structure of this book. When preparing for the workshop, I realized that a discrete-time system offers a natural setting for the development of predictive control, and the design framework is easier to understand when it is presented in discrete time. The discussion with Mr Oliver Jackson (Editor, Springer) in (ACC) 2007 also helped me think harder on how to deliver the complex issues in a manner as simple as possible. In the end, I chose to present the materials in a bottom-up manner to suit the background of a fourth-year undergraduate student. The best way to do so is to actually try the presentation on a class of fourth year students and then find out what the difficult issues are. This

is indeed what I have tried. The first three chapters were taught in the classroom, and the students provided the suggestions and feedback, which helped to set the level and pace of the presentation. The book is intended for readers who have completed or are about to complete four years engineering studies with some basic knowledge in state-space design. The textbooks for an introduction to state-space methods include Franklin *et al.* (1991), Kailath (1980), Bay (1999) and Goodwin *et al.* (2000). However, some of the basic knowledge will be reviewed in the relevant chapters, and the book is self-contained with MATLAB tutorials and numerous examples. The targeted readers are students, practitioners, instructors and researchers who wish to apply predictive control.

Book Structure

The structure of the book is illustrated by the block diagram as shown in Figure 0.1. There are ten chapters in this book, and both continuous-time and discrete-time predictive control systems are introduced and discussed (see Chapters 1 to 8). Discrete-time algorithms are introduced first because of their strong relevance to industrial applications and their natural settings in the development. Chapters 9 and 10 contain both continuous-time and discrete-time systems, from using special state-space realization to implementation of those control systems via MATLAB Real-time Workshop and SIMULINK xPC Target. Each chapter contains MATLAB tutorials, which illustrate how the algorithms can be used in simulation, computation and implementation, and a problem section for practice of the design.

Chapter 1 is for beginners, where we will use simple examples to show the principle of receding horizon control, which underpins predictive control. The solutions are limited to simple analytical solutions. In Chapter 1, we will also discuss implementation using observer and observability with several simple examples. In Chapter 2, the basic ideas about constrained control are introduced within the framework of receding horizon control. The key is to formulate the constrained control problem as a real-time optimization problem subject to inequality constraints. The solution to this problem relies on application of a quadratic programming procedure. Given that the majority of the readers may not have studied constrained optimization before, in this chapter, we will also discuss optimization with equality and inequality constraints. The discussion is followed by the introduction of Hildreth's quadratic programming procedure, which offers simplicity and reliability in real-time implementation.

In essence, the core technique in the design of discrete-time MPC is based on optimizing the future control trajectory subject to plant operational constraints. In the traditional predictive control, as demonstrated in Chapter 1 and 2, by assuming a finite control horizon N_c and prediction horizon N_p , the difference of the control signal $\Delta u(k)$ for $k = 0, 1, 2, \dots, N_c - 1$ is captured by the control vector ΔU , while the rest of the $\Delta u(k)$ for $k = N_c, N_c + 1, \dots, N_p$

is assumed to be zero. The idea in Chapter 3 is to generalize the traditional design procedure by introducing a set of discrete Laguerre functions into the design. This generalization will help us in re-formulating the predictive control problem and simplifying the solutions, in addition to providing a set of new performance-tuning parameters that can be readily understood by engineers. Furthermore, a long control horizon can be realized through the exponential nature of the Laguerre functions, hence without using a large number of parameters. Several MATLAB tutorials are presented in this chapter for the design of discrete-time predictive control systems, with or without constraints.

It is fair to say that the majority of industrial control systems require integral action. In this book, the design models are embedded with integrators to achieve this objective, which is similar to other classical predictive control systems. Because of the embedded integrators, the prediction horizon N_p as a design parameter plays an important role in a predictive control system. In Chapter 4, we demonstrate that if it is chosen too short, the closed-loop

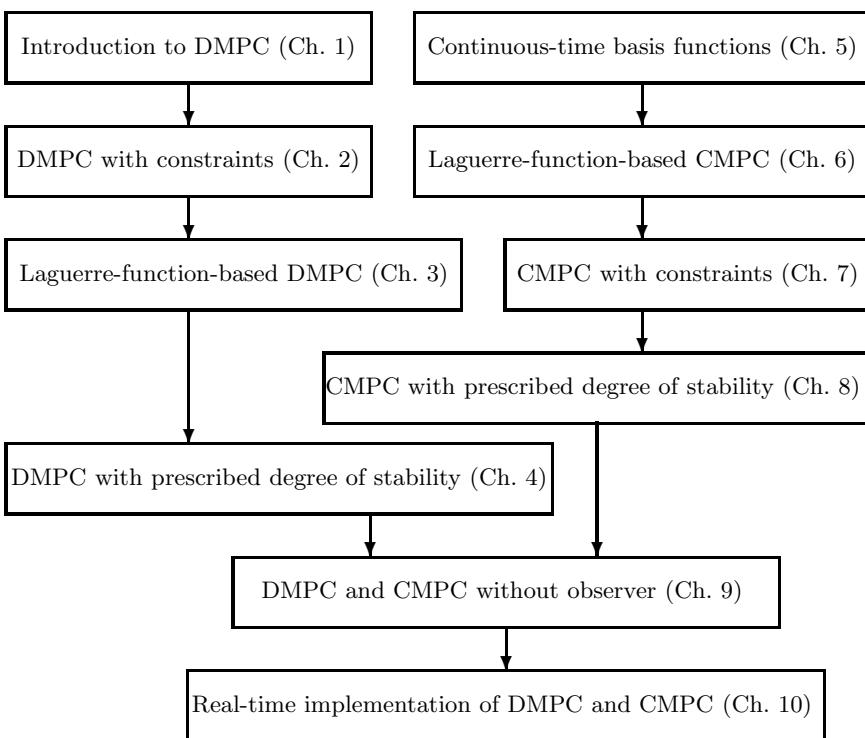


Fig. 0.1. Structure of the book

system is not necessarily stable, and if it is too large, the predictive control system will encounter a numerical stability problem. To overcome this problem, in Chapter 4, we propose the use of an exponentially weighted moving horizon window in model predictive control design. Within this framework, the numerical ill-conditioning problem is resolved by a simple modification of the design model. Equally important are an asymptotic stability result for predictive control designed using infinite horizon with exponential data weighting and a modification of the weight matrices, as well as a result that establishes the predictive control system with a prescribed degree of stability. Analytical and numerical results are used in this chapter to show the equivalence between the new class of discrete-time predictive control systems and the classical discrete-time linear quadratic regulators (DLQR) without constraints. When constraints are present, the optimal solutions are obtained by minimizing the exponentially weighted cost subject to transformed inequality constraints.

Chapters 6 to 8 will introduce the continuous-time predictive control results. To prepare the background, in Chapter 5, we will discuss continuous-time orthonormal basis functions and their applications in dynamic system modelling. Laguerre functions and Kautz functions are special classes of orthonormal basis functions. Both sets of functions possess simple Laplace transforms, and can be compactly represented by state-space models. The key property is that when using the orthonormal functions, modelling of the impulse response of a stable system, which has a bounded integral squared value, will have a guaranteed convergence as the number of terms used increases. This forms the fundamental principle of the model predictive control design methods presented in this book.

After introducing the background information, in Chapter 6, we begin with the topics in continuous-time model predictive control (CMPC). It is natural that when the design model is embedded with integrators, the derivative of the control signal should be modelled by the orthonormal basis functions, not the control signal itself. With this as a start point, systematically, we will cover the principles of continuous-time predictive control design, and the solutions of the optimal control problem. It shows that when constraints are not involved in the design, the continuous-time model predictive control scheme becomes a state feedback control system, with the gain being chosen from minimizing a finite prediction horizon cost. The continuous-time Laguerre functions and Kautz functions discussed in Chapter 5 are utilized in the design of continuous-time model predictive control.

In Chapter 7, we introduce continuous-time model predictive control with constraints. Similar to the discrete-time case, we will first formulate the constraints for the continuous-time predictive control system, and present the numerical solution of the constrained control problem using a quadratic programming procedure. Because of the nature of the continuous-time formulation such as fast sampling, there might be computational delays when a quadratic programming procedure is used in the solution of the real-time op-

timization problem. In general terms, we discuss the real-time implementation of continuous-time model predictive control in the presence of constraints in this chapter too.

The numerical instability problem discussed in the discrete-time case, also occurs in the continuous-time algorithms that is shown by a numerical example in Chapter 8. In a similar spirit to previous chapters, Chapter 8 proposes the use of exponential data weighting in the design of continuous-time model predictive control systems. This essentially transforms the original state and derivative of the control variables into exponentially weighted variables for the optimization procedure. With a simple modification on the weight matrices, asymptotic stability is established for model predictive control systems with infinite prediction horizon. Similarly, a prescribed degree of stability can also be obtained. Without constraints, analytical and numerical results are used to demonstrate the equivalence between the continuous-time model predictive controllers and the classical linear quadratic regulators (LQR). Constraints are imposed on the transformed variables.

In a general framework of state feedback control, an observer is often needed for its implementation. The design of an observer is a separate task from the design of predictive controller. The role of an observer is to ensure small errors between the estimated and the actual state variables. However, if one faces many inputs and many outputs in a system, tuning of an observer's dynamics may not be a trivial task. The classical predictive control systems, such as dynamic matrix control (DMC) and generalized predictive control (GPC), have directly utilized plant input and output signals in the closed-loop feedback control, hence avoiding observers in their implementation. In Chapter 9, we will link the predictive control systems designed using the framework of state space to the classical predictive control systems. The key to the link is to choose the state variables to be identical to the feedback variables that have been used in the classical predictive control systems. Once the state-space model is formulated, the framework from the previous chapters is naturally extended to the classical predictive control systems, preserving all the advantages of a state-space design, including stability analysis, exponential data weighting and LQR equivalence. In addition, because of the utilization of plant input and output signals in the implementation, the predictive controller can be represented in a transfer function form, allowing a direct frequency response analysis of the system to obtain critical information, such as gain and phase margins. It is worthwhile to point out that the discrete-time single-input and single-output predictive controller is very simple and easy for implementation. Simplicity as a feature of the algorithms remains when they are extended to multi-input and multi-output systems. Different from discrete-time, in the continuous-time design, an implementation filter is required, and the poles of the filter become part of the desired closed-loop poles when we choose to optimize the output errors in the design.

Chapter 10 presents three different implementation procedures for model predictive control systems. The first implementation is based on a (low-cost)

micro-controller for controlling a DC motor. In this application, the MATLAB design programs are utilized to calculate the predictive controller gain and the previous MATLAB closed-loop simulation program is converted to a C program for real-time implementation on the micro-controller. The second implementation is based on MATLAB Real-time Workshop and xPC target. This application is very useful for those who are not familiar with C language because the MATLAB Real-time Workshop and xPC target perform the conversion from MATLAB programs to C programs through their compilers in a systematic way. With these tools, we only need to create MATLAB embedded functions for the real-time applications. The third implementation uses the platform of a real-time PC-based supervisory control and data acquisition (SCADA) system. A pilot food extrusion plant is controlled by the continuous-time predictive controller developed in Chapter 6. The previous MATLAB closed-loop simulation program for a continuous-time system is converted to C program for this real-time implementation.

Acknowledgements

There are two aspects that have motivated me in the writing of this book. The excitement of fitting together the work I have been doing over the years into a book is my first motivation. The second is that I need to express my gratitude to those who helped me over the years. With this book, I have the opportunity to formally thank so many colleagues and friends.

My first gratitude goes to Professor Graham C. Goodwin who was the Director of the Center for Integrated Dynamics and Control, University of Newcastle, Australia where I worked from February 1998 to February 2002. His encouragement and help since November 1997 are gratefully acknowledged. My special thanks go to Dr Robert Snow, who was previously the Dean of Engineering, RMIT University, Australia and who had helped me tremendously during my early years in RMIT.

My thanks go to Professor Michael Johnson who is a joint editor of the Advances in Industrial Control Book series by Springer. Professor Michael Johnson provided me with the opportunity to write this book and very valuable feedback towards its improvement. His encouragement is very much appreciated.

Over the years, numerous people have helped me and worked with me in various stages of my career. I would like to thank Professor Peter Gawthrop (Glasgow, UK), Professor David Owens (Sheffield, U.K), Professor Eric Rogers (Southampton, UK), Professor Peter Young (Lancaster, UK), Professor Hugues Garnier (Nancy, France), Professor Alex Penlidis (Waterloo, Canada), Professor Rick Middleton (Newcastle, Australia), Professor Xinghou Yu (RMIT

University, Australia), Dr Anthony Rossiter (Sheffield, UK), Dr Charlie Chesarri (Food Science Australia), Mr John Keone and Mr Pat Borland (ANCA, Australia) for their help and support.

My thanks go to my current and former students, Dr Nguyen Vu Truong, Stephen Smith, Dae Yoo, Stephen Gale, Jianming Huang, Noor Azizi Mardi, Liwei Ma, Shan Cai and Yang Huang who carried out the real-time implementation of predictive control on various systems. My thanks also go to the students in the class of EEET 2221 (2007) and EEET 2223 (2007) in RMIT for whom I designed the first three chapters of this book. Dr Paul Riseborough (BAE Systems, Australia) has given me valuable feedback towards improvement of this book.

I gratefully acknowledge helpful suggestions and assistance by Oliver Jackson (Editor, Springer) and his staff.

Finally, I thank my husband, Dr Jianshe Guan, for his support over the years from Britain to Canada, then from Canada to Australia, and my son, Robin P. Guan, for his love and affection.

Melbourne, Australia, October 2008

Liuping Wang

Contents

List of Symbols and Abbreviations	xxvii
1 Discrete-time MPC for Beginners	1
1.1 Introduction	1
1.1.1 Day-to-day Application Example of Predictive Control ..	1
1.1.2 Models Used in the Design	3
1.2 State-space Models with Embedded Integrator	4
1.2.1 Single-input and Single-output System	4
1.2.2 MATLAB Tutorial: Augmented Design Model	6
1.3 Predictive Control within One Optimization Window	7
1.3.1 Prediction of State and Output Variables	7
1.3.2 Optimization	9
1.3.3 MATLAB Tutorial: Computation of MPC Gains	13
1.4 Receding Horizon Control	15
1.4.1 Closed-loop Control System	16
1.4.2 MATLAB Tutorial: Implementation of Receding Horizon Control	20
1.5 Predictive Control of MIMO Systems	22
1.5.1 General Formulation of the Model	22
1.5.2 Solution of Predictive Control for MIMO Systems	26
1.6 State Estimation	27
1.6.1 Basic Ideas About an Observer	28
1.6.2 Basic Results About Observability	30
1.6.3 Kalman Filter	33
1.6.4 Tuning Observer Dynamics	34
1.7 State Estimate Predictive Control	34
1.8 Summary	37
Problems	39

2 Discrete-time MPC with Constraints	43
2.1 Introduction	43
2.2 Motivational Examples	43
2.3 Formulation of Constrained Control Problems	47
2.3.1 Frequently Used Operational Constraints	47
2.3.2 Constraints as Part of the Optimal Solution	50
2.4 Numerical Solutions Using Quadratic Programming	53
2.4.1 Quadratic Programming for Equality Constraints	53
2.4.2 Minimization with Inequality Constraints	58
2.4.3 Primal-Dual Method	62
2.4.4 Hildreth's Quadratic Programming Procedure	63
2.4.5 MATLAB Tutorial: Hildreth's Quadratic Programming	67
2.4.6 Closed-form Solution of λ^*	68
2.5 Predictive Control with Constraints on Input Variables	69
2.5.1 Constraints on Rate of Change	70
2.5.2 Constraints on Amplitude of the Control	73
2.5.3 Constraints on Amplitude and Rate of Change	77
2.5.4 Constraints on the Output Variable	78
2.6 Summary	81
Problems	83
3 Discrete-time MPC Using Laguerre Functions	85
3.1 Introduction	85
3.2 Laguerre Functions and DMPC	85
3.2.1 Discrete-time Laguerre Networks	86
3.2.2 Use of Laguerre Networks in System Description	90
3.2.3 MATLAB Tutorial: Use of Laguerre Functions in System Modelling	90
3.3 Use of Laguerre Functions in DMPC Design	92
3.3.1 Design Framework	93
3.3.2 Cost Functions	94
3.3.3 Minimization of the Objective Function	97
3.3.4 Convolution Sum	98
3.3.5 Receding Horizon Control	98
3.3.6 The Optimal Trajectory of Incremental Control	99
3.4 Extension to MIMO Systems	106
3.5 MATLAB Tutorial Notes	108
3.5.1 DMPC Computation	108
3.5.2 Predictive Control System Simulation	115
3.6 Constrained Control Using Laguerre Functions	118
3.6.1 Constraints on the Difference of the Control Variable	118
3.6.2 Constraints on the Amplitudes of the Control Signal	121
3.7 Stability Analysis	127
3.7.1 Stability with Terminal-State Constraints	127
3.7.2 Stability with Large Prediction Horizon	129

3.8	Closed-form Solution of Constrained Control for SISO Systems	131
3.8.1	MATLAB Tutorial: Constrained Control of DC Motor	135
3.9	Summary	143
	Problems	144
4	Discrete-time MPC with Prescribed Degree of Stability	149
4.1	Introduction	149
4.2	Finite Prediction Horizon: Re-visited	149
4.2.1	Motivational Example	150
4.2.2	Origin of the Numerical Conditioning Problem	150
4.3	Use of Exponential Data Weighting	152
4.3.1	The Cost Function	152
4.3.2	Optimization of Exponentially Weighted Cost Function	153
4.3.3	Interpretation of Results from Exponential Weighting	156
4.4	Asymptotic Closed-loop Stability with Exponential Weighting	158
4.4.1	Modification of Q and R Matrices	158
4.4.2	Interpretation of the Results	160
4.5	Discrete-time MPC with Prescribed Degree of Stability	165
4.6	Tuning Parameters for Closed-loop Performance	170
4.6.1	The Relationship Between P_∞ and J_{min}	171
4.6.2	Tuning Procedure Once More	176
4.7	Exponentially Weighted Constrained Control	179
4.8	Additional Benefit	182
4.9	Summary	186
	Problems	188
5	Continuous-time Orthonormal Basis Functions	193
5.1	Introduction	193
5.2	Orthonormal Expansion	193
5.3	Laguerre Functions	194
5.4	Approximating Impulse Responses	197
5.5	Kautz Functions	202
5.5.1	Kautz Functions in the Time Domain	204
5.5.2	Modelling the System Impulse Response	205
5.6	Summary	206
	Problems	207
6	Continuous-time MPC	209
6.1	Introduction	209
6.2	Model Structures for CMPC Design	209
6.2.1	Model Structure	211
6.2.2	Controllability and Observability of the Model	215
6.3	Model Predictive Control Using Finite Prediction Horizon	216
6.3.1	Modelling the Control Trajectory	217
6.3.2	Predicted Plant Response	218

6.3.3	Analytical Solution of the Predicted Response	219
6.3.4	The Recursive Solution.....	221
6.4	Optimal Control Strategy	224
6.5	Receding Horizon Control	227
6.6	Implementation of the Control Law in Digital Environment ..	234
6.6.1	Estimation of the States.....	234
6.6.2	MATLAB Tutorial: Closed-loop Simulation	237
6.7	Model Predictive Control Using Kautz Functions	240
6.8	Summary.....	244
	Problems	245
7	Continuous-time MPC with Constraints	249
7.1	Introduction	249
7.2	Formulation of the Constraints	249
7.2.1	Frequently Used Constraints	249
7.2.2	Constraints as Part of the Optimal Solution	251
7.3	Numerical Solutions for the Constrained Control Problem ..	257
7.4	Real-time Implementation of Continuous-time MPC	262
7.5	Summary.....	266
	Problems	267
8	Continuous-time MPC with Prescribed Degree of Stability 271	
8.1	Introduction	271
8.2	Motivating Example	271
8.3	CMPC Design Using Exponential Data Weighting	274
8.4	CMPC with Asymptotic Stability	277
8.5	Continuous-time MPC with Prescribed Degree of Stability ..	283
8.5.1	The Original Anderson and Moore's Results	283
8.5.2	CMPC with a Prescribed Degree of Stability	284
8.5.3	Tuning Parameters and Design Procedure	286
8.6	Constrained Control with Exponential Data Weighting	288
8.7	Summary.....	291
	Problems	293
9	Classical MPC Systems in State-space Formulation	297
9.1	Introduction	297
9.2	Generalized Predictive Control in State-space Formulation ..	298
9.2.1	Special Class of Discrete-time State-space Structures ..	298
9.2.2	General NMSS Structure for GPC Design	301
9.2.3	Generalized Predictive Control in State-space Formulation	302
9.3	Alternative Formulation to GPC	305
9.3.1	Alternative Formulation for SISO Systems	305
9.3.2	Closed-loop Poles of the Predictive Control System ..	307
9.3.3	Transfer Function Interpretation	310

9.4	Extension to MIMO Systems	313
9.4.1	MNSS Model for MIMO Systems	314
9.4.2	Case Study of NMSS Predictive Control System	315
9.5	Continuous-time NMSS model	320
9.6	Case Studies for Continuous-time MPC	323
9.7	Predictive Control Using Impulse Response Models	326
9.8	Summary	329
	Problems	330
10	Implementation of Predictive Control Systems	333
10.1	Introduction	333
10.2	Predictive Control of DC Motor Using a Micro-controller	333
10.2.1	Hardware Configuration	334
10.2.2	Model Development	336
10.2.3	DMPC Tuning	337
10.2.4	DMPC Implementation	338
10.2.5	Experimental Results	339
10.3	Implementation of Predictive Control Using xPC Target	340
10.3.1	Overview	340
10.3.2	Creating a SIMULINK Embedded Function	342
10.3.3	Constrained Control of DC Motor Using xPC Target	347
10.4	Control of Magnetic Bearing Systems	349
10.4.1	System Identification	351
10.4.2	Experimental Results	352
10.5	Continuous-time Predictive Control of Food Extruder	353
10.5.1	Experimental Setup	355
10.5.2	Mathematical Models	357
10.5.3	Operation of the Model Predictive Controller	358
10.5.4	Controller Tuning Parameters	359
10.5.5	On-line Control Experiments	360
10.6	Summary	365
	References	367
	Index	373

List of Symbols and Abbreviations

Symbols

a	Scaling factor for discrete-time Laguerre functions
$\arg \min$	Minimizing argument
A	State matrix of state-space model
B	Input-to-state matrix of state-space model
C	State-to-output matrix of state-space model
D	Direct feed-through matrix of state-space model
(A, B, C, D)	State-space realization
ΔU	Parameter vector for the control sequence
$\Delta u(k_i + m)$	future incremental control at sample m
$\Delta u^{\min}, \Delta u^{\max}$	Minimum and maximum limits for Δu
F, Φ	Pair of matrices used in the prediction equation $Y = Fx(k_i) + \Phi\Delta U$
$G(s)$	Transfer function model
$I_{q \times q}$	Identity matrix with appropriate dimensions
J	Performance index for optimization
K_{lqr}	Feedback control gain using LQR
K_{mpc}	Feedback control gain using MPC
K_x	State feedback control gain vector related to $\Delta x_m(\cdot)$ or $\dot{x}_m(\cdot)$
K_y	State feedback control gain related to y
K_{ob}	Observer gain vector
$\kappa(A)$	Condition number of A matrix
$l_i(\cdot)$	The i th discrete or continuous-time Laguerre function
$L(\cdot)$	Discrete and continuous-time Laguerre functions in vector form
$L_i(s)$	Laplace transform of the i th continuous-time Laguerre function
$L_i(z)$	z-transform of the i th discrete Laguerre function
λ	Lagrange multiplier

$\lambda_i(A)$	The i th eigenvalue of matrix A
m	Number of inputs, also the m th future sample in discrete time
M, γ	Pair of matrix, vector for inequality constraints ($Mx \leq \gamma$)
N	Number of terms used in Laguerre function expansion, both continuous and discrete time
N_c	Control horizon
N_p	Prediction horizon
o_m	Zero vector with appropriate dimension
o_k	Zero row vector ($k = 1, 2, \dots$) with appropriate dimensions
$o_{q \times q}$	$q \times q$ zero matrix
$o_{q \times m}$	$q \times m$ zero matrix
Ω, Ψ	Pair of matrices in the cost of predictive control $J = \eta^T \Omega \eta + 2\eta^T \Psi x(\cdot) + cons$
η	Parameter vector in the Laguerre expansion
p	Scaling factor for continuous-time Laguerre functions
Q, R	Pair of weight matrices in the cost function of predictive control
q^{-i}	Backward shift operator, $q^{-i}[f(k)] = f(k - i)$
q	Number of outputs
$r(\cdot)$	Set-point signal
S_{act}	Index set of active constraints
T_p	Prediction horizon in continuous-time
$u(\cdot)$	Control signal
u^{min}, u^{max}	Minimum and maximum limits for u
$x(\cdot)$	State variable
$x(k_i + m \mid k_i)$	Predicted state variable vector at sample time m , given current state $x(k_i)$
$x(t_i + \tau \mid t_i)$	Predicted state variable vector at time τ given current state $x(t_i)$
$\hat{x}(\cdot)$	Estimated state variable vector in both continuous and discrete-time
$y(\cdot)$	Output signal
Y	Predicted output data vector
y^{min}, y^{max}	Minimum and maximum limits for y

Abbreviations

CMPC	Continuous-time model predictive control
DLQR	Discrete-time linear quadratic regulator

DMPC	Discrete-time model predictive control
FIR	Finite impulse response
LQR	Linear quadratic regulator
MIMO	Multiple-input, multiple-output
SISO	Single-input, single-output

Discrete-time MPC for Beginners

1.1 Introduction

In this chapter, we will introduce the basic ideas and terms about model predictive control. In Section 1.2, a single-input and single-output state-space model with an embedded integrator is introduced, which is used in the design of discrete-time predictive controllers with integral action in this book. In Section 1.3, we examine the design of predictive control within one optimization window. This is demonstrated by simple analytical examples. With the results obtained from the optimization, in Section 1.4, we discuss the ideas of receding horizon control, and state feedback gain matrices, and the closed-loop configuration of the predictive control system. The results are extended to multi-input and multi-output systems in Section 1.5. In a general framework of state-space design, an observer is needed in the implementation, and this is discussed in Section 1.6. With a combination of estimated state variables and the predictive controller, in Section 1.7, we present state estimate predictive control.

1.1.1 Day-to-day Application Example of Predictive Control

The general design objective of model predictive control is to compute a trajectory of a future manipulated variable u to optimize the future behaviour of the plant output y . The optimization is performed within a limited time window by giving plant information at the start of the time window. To help understand the basic ideas that have been used in the design of predictive control, we examine a typical planning activity of our day-to-day work.

The day begins at 9 o'clock in the morning. We are, as a team, going to complete the tasks of design and implementation of a model predictive control system for a liquid vessel. The rule of the game is that we always plan our activities for the next 8 hours work, however, we only implement the plan for the first hour. This planning activity is repeated for every fresh hour until the tasks are completed.

Given the amount of background work that we have completed for 9 o'clock, we plan ahead for the next 8 hours. Assume that the work tasks are divided into modelling, design, simulation and implementation. Completing these tasks will be a function of various factors, such as how much effort we will put in, how well we will work as a team and whether we will get some additional help from others. **These are the manipulated variables in the planning problem.** Also, we have our limitations, such as our ability to understand the design problem, and whether we have good skills of computer hardware and software engineering. **These are the hard and soft constraints in the planning.** **The background information we have already acquired is paramount for this planning work.**

After everything is considered, we determine the design tasks for the next 8 hours as functions of the manipulated variables. Then we calculate hour-by-hour what we need to do in order to complete the tasks. In this calculation, **based on the background information, we will take our limitations into consideration as the constraints,** and **find the best way to achieve the goal.** The end result of this planning gives us our projected activities from 9 o'clock to 5 o'clock. Then we start working by implementing the activities for the first hour of our plan.

At 10 o'clock, we check how much we have actually done for the first hour. This information is used for the planning of our next phase of activities. Maybe we have done less than we planned because we could not get the correct model or because one of the key members went for an emergency meeting. Nevertheless, at 10 o'clock, we make an assessment on what we have achieved, and use this updated information for planning our activities for the next 8 hours. Our objective may remain the same or may change. The length of time for the planning remains the same (8 hours). We repeat the same planning process as it was at 9 o'clock, which then gives us the new projected activities for the next 8 hours. We implement the first hour of activities at 10 o'clock. Again at 11 o'clock, we assess what we have achieved again and use the updated information for the plan of work for the next 8 hours. The planning and implementation process is repeated every hour until the original objective is achieved.

There are **three key elements required in the planning.** The first is the way of predicting **what might happen** (model); the second is the instrument of **assessing our current activities** (measurement); and the third is the instrument of **implementing the planned activities** (realization of control). The key issues in the planning exercise are:

1. the **time window** for the planning is fixed at 8 hours;
2. we need to know our current status before the planning;
3. we take the best approach for the 8 hours work by taking the constraints into consideration, and the optimization is performed in real-time with a **moving horizon** time window and with the latest information available.

The planning activity described here involves the principle of MPC. In this example, it is described by the terms that are to be used frequently in the

following: the moving horizon window, prediction horizon, receding horizon control, and control objective. They are introduced as below.

1. **Moving horizon window**: the time-dependent window from an arbitrary time t_i to $t_i + T_p$. The length of the window T_p remains constant. In this example, the planning activity is performed within an 8-hour window, thus $T_p = 8$, with the measurement taken every hour. However, t_i , which defines the beginning of the optimization window, increases on an hourly basis, starting with $t_i = 9$.
2. **Prediction horizon**: dictates how ‘far’ we wish the future to be predicted for. This parameter equals the length of the moving horizon window, T_p .
3. **Receding horizon control**: although the optimal trajectory of future control signal is completely described within the moving horizon window, the actual control input to the plant only takes the first sample of the control signal, while neglecting the rest of the trajectory.
4. In the **planning process**, we need the information at time t_i in order to predict the future. This information is denoted as $x(t_i)$ which is a vector containing many relevant factors, and is either directly measured or estimated.
5. A given model that will describe the dynamics of the system is paramount in predictive control. A good dynamic model will give a consistent and accurate prediction of the future.
6. In order to make the best decision, a criterion is needed to reflect the objective. The objective is related to an error function based on the difference between the desired and the actual responses. This objective function is often called the cost function J , and the optimal control action is found by minimizing this cost function within the optimization window.

1.1.2 Models Used in the Design

There are three general approaches to predictive control design. Each approach uses a unique model structure. In the earlier formulation of model predictive control, finite impulse response (FIR) models and step response models were favoured. FIR model/step response model based design algorithms include dynamic matrix control (DMC) (Cutler and Ramaker, 1979) and the quadratic DMC formulation of Garcia and Morshedi (1986). The FIR type of models are appealing to process engineers because the model structure gives a transparent description of process time delay, response time and gain. However, they are limited to stable plants and often require large model orders. This model structure typically requires 30 to 60 impulse response coefficients depending on the process dynamics and choice of sampling intervals. Transfer function models give a more parsimonious description of process dynamics and are applicable to both stable and unstable plants. Representatives of transfer function model-based predictive control include the predictive control algorithm of Peterka (Peterka, 1984) and the generalized

predictive control (GPC) algorithm of Clarke and colleagues (Clarke *et al.*, 1987). The transfer function model-based predictive control is often considered to be less effective in handling multivariable plants. A state-space formulation of GPC has been presented in Ordys and Clarke (1993).

Recent years have seen the growing popularity of predictive control design using state-space design methods (Ricker, 1991, Rawlings and Muske, 1993, Rawlings, 2000, Maciejowski, 2002). In this book, we will use state-space models, both in continuous time and discrete time for simplicity of the design framework and the direct link to the classical linear quadratic regulators.

1.2 State-space Models with Embedded Integrator

Model predictive control systems are designed based on a mathematical model of the plant. The model to be used in the control system design is taken to be a state-space model. By using a state-space model, the current information required for predicting ahead is represented by the state variable at the current time.

1.2.1 Single-input and Single-output System

For simplicity, we begin our study by assuming that the underlying plant is a single-input and single-output system, described by:

$$x_m(k+1) = A_m x_m(k) + B_m u(k), \quad (1.1)$$

$$y(k) = C_m x_m(k), \quad (1.2)$$

where u is the manipulated variable or input variable; y is the process output; and x_m is the state variable vector with assumed dimension n_1 . Note that this plant model has $u(k)$ as its input. Thus, we need to change the model to suit our design purpose in which an integrator is embedded.

Note that a general formulation of a state-space model has a direct term from the input signal $u(k)$ to the output $y(k)$ as

$$y(k) = C_m x_m(k) + D_m u(k).$$

However, due to the principle of receding horizon control, where a current information of the plant is required for prediction and control, we have implicitly assumed that the input $u(k)$ cannot affect the output $y(k)$ at the same time. Thus, $D_m = 0$ in the plant model.

Taking a difference operation on both sides of (1.1), we obtain that

$$x_m(k+1) - x_m(k) = A_m(x_m(k) - x_m(k-1)) + B_m(u(k) - u(k-1)).$$

Let us denote the difference of the state variable by

$$\Delta x_m(k+1) = x_m(k+1) - x_m(k); \quad \Delta x_m(k) = x_m(k) - x_m(k-1),$$

and the difference of the control variable by

$$\Delta u(k) = u(k) - u(k-1).$$

These are the increments of the variables $x_m(k)$ and $u(k)$. With this transformation, the difference of the state-space equation is:

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k). \quad (1.3)$$

Note that the input to the state-space model is $\Delta u(k)$. The next step is to connect $\Delta x_m(k)$ to the output $y(k)$. To do so, a new state variable vector is chosen to be

$$x(k) = [\Delta x_m(k)^T \ y(k)]^T,$$

where superscript T indicates matrix transpose. Note that

$$\begin{aligned} y(k+1) - y(k) &= C_m(x_m(k+1) - x_m(k)) = C_m \Delta x_m(k+1) \\ &= C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k). \end{aligned} \quad (1.4)$$

Putting together (1.3) with (1.4) leads to the following state-space model:

$$\begin{aligned} \underbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}_{x(k+1)} &= \underbrace{\begin{bmatrix} A_m & o_m^T \\ C_m A_m & 1 \end{bmatrix}}_A \underbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}_{x(k)} + \underbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}_B \Delta u(k) \\ y(k) &= \underbrace{\begin{bmatrix} o_m & 1 \end{bmatrix}}_C \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}, \end{aligned} \quad (1.5)$$

where $o_m = \overbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \end{bmatrix}}^{n_1}$. The triplet (A, B, C) is called the augmented model, which will be used in the design of predictive control.

Example 1.1. Consider a discrete-time model in the following form:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k) \end{aligned} \quad (1.6)$$

where the system matrices are

$$A_m = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}; \quad B_m = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}; \quad C_m = [1 \ 0].$$

Find the triplet matrices (A, B, C) in the augmented model (1.5) and calculate the eigenvalues of the system matrix, A , of the augmented model.

Solution. From (1.5), $n_1 = 2$ and $o_m = [0 \ 0]$. The augmented model for this plant is given by

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k), \end{aligned} \quad (1.7)$$

where the augmented system matrices are

$$A = \begin{bmatrix} A_m & o_m^T \\ C_m A_m & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}; \quad B = \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1 \\ 0.5 \end{bmatrix};$$

$$C = [o_m \ 1] = [0 \ 0 \ 1].$$

The characteristic equation of matrix A is given by

$$\begin{aligned} \rho(\lambda) &= \det(\lambda I - A) = \det \begin{bmatrix} \lambda I - A_m & o_m^T \\ -C_m A_m & (\lambda - 1) \end{bmatrix} \\ &= (\lambda - 1) \det(\lambda I - A_m) = (\lambda - 1)^3. \end{aligned} \quad (1.8)$$

Therefore, the augmented state-space model has three eigenvalues at $\lambda = 1$. Among them, two are from the original integrator plant, and one is from the augmentation of the plant model.

1.2.2 MATLAB Tutorial: Augmented Design Model

Tutorial 1.1. *The objective of this tutorial is to demonstrate how to obtain a discrete-time state-space model from a continuous-time state-space model, and form the augmented discrete-time state-space model. Consider a continuous-time system has the state-space model:*

$$\begin{aligned} \dot{x}_m(t) &= \begin{bmatrix} 0 & 1 & 0 \\ 3 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} x_m(t) + \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix} u(t) \\ y(t) &= [0 \ 1 \ 0] x_m(t). \end{aligned} \quad (1.9)$$

Step by Step

1. Create a new file called `extmodel.m`. We form a continuous-time state variable model; then this continuous-time model is discretized using MATLAB function ‘`c2dm`’ with specified sampling interval Δt .
2. Enter the following program into the file:

```
Ac = [0 1 0; 3 0 1; 0 1 0];
Bc= [1; 1; 3];
Cc=[0 1 0];
Dc=zeros(1,1);
Delta_t=1;
[Ad,Bd,Cd,Dd]=c2dm(Ac,Bc,Cc,Dc,Delta_t);
```

3. The dimensions of the system matrices are determined to discover the numbers of states, inputs and outputs. The augmented state-space model is produced. Continue entering the following program into the file:

```
[m1,n1]=size(Cd);
[n1,n_in]=size(Bd);
A_e=eye(n1+m1,n1+m1);
A_e(1:n1,1:n1)=Ad;
A_e(n1+1:n1+m1,1:n1)=Cd*Ad;
B_e=zeros(n1+m1,n_in);
B_e(1:n1,:)=Bd;
B_e(n1+1:n1+m1,:)=Cd*Bd;
C_e=zeros(m1,n1+m1);
C_e(:,n1+1:n1+m1)=eye(m1,m1);
```

4. Run this program to produce the augmented state variable model for the design of predictive control.

1.3 Predictive Control within One Optimization Window

Upon formulation of the mathematical model, the next step in the design of a predictive control system is to calculate the predicted plant output with the future control signal as the adjustable variables. This prediction is described within an optimization window. This section will examine in detail the optimization carried out within this window. Here, we assume that the current time is k_i and the length of the optimization window is N_p as the number of samples. For simplicity, the case of single-input and single-output systems is considered first, then the results are extended to multi-input and multi-output systems.

1.3.1 Prediction of State and Output Variables

Assuming that at the sampling instant k_i , $k_i > 0$, the state variable vector $x(k_i)$ is available through measurement, the state $x(k_i)$ provides the current plant information. The more general situation where the state is not directly measured will be discussed later. The future control trajectory is denoted by

$$\Delta u(k_i), \Delta u(k_i + 1), \dots, \Delta u(k_i + N_c - 1),$$

where N_c is called the control horizon dictating the number of parameters used to capture the future control trajectory. With given information $x(k_i)$, the future state variables are predicted for N_p number of samples, where N_p is called the prediction horizon. N_p is also the length of the optimization window. We denote the future state variables as

$$x(k_i + 1 | k_i), x(k_i + 2 | k_i), \dots, x(k_i + m | k_i), \dots, x(k_i + N_p | k_i),$$

where $x(k_i + m \mid k_i)$ is the predicted state variable at $k_i + m$ with given current plant information $x(k_i)$. The control horizon N_c is chosen to be less than (or equal to) the prediction horizon N_p .

Based on the state-space model (A, B, C) , the future state variables are calculated sequentially using the set of future control parameters:

$$\begin{aligned} x(k_i + 1 \mid k_i) &= Ax(k_i) + B\Delta u(k_i) \\ x(k_i + 2 \mid k_i) &= Ax(k_i + 1 \mid k_i) + B\Delta u(k_i + 1) \\ &= A^2x(k_i) + AB\Delta u(k_i) + B\Delta u(k_i + 1) \\ &\vdots \\ x(k_i + N_p \mid k_i) &= A^{N_p}x(k_i) + A^{N_p-1}B\Delta u(k_i) + A^{N_p-2}B\Delta u(k_i + 1) \\ &\quad + \dots + A^{N_p-N_c}B\Delta u(k_i + N_c - 1). \end{aligned}$$

From the predicted state variables, the predicted output variables are, by substitution

$$\begin{aligned} y(k_i + 1 \mid k_i) &= CAx(k_i) + CB\Delta u(k_i) & (1.10) \\ y(k_i + 2 \mid k_i) &= CA^2x(k_i) + CAB\Delta u(k_i) + CB\Delta u(k_i + 1) \\ y(k_i + 3 \mid k_i) &= CA^3x(k_i) + CA^2B\Delta u(k_i) + CAB\Delta u(k_i + 1) \\ &\quad + CB\Delta u(k_i + 2) \\ &\vdots \\ y(k_i + N_p \mid k_i) &= CA^{N_p}x(k_i) + CA^{N_p-1}B\Delta u(k_i) + CA^{N_p-2}B\Delta u(k_i + 1) \\ &\quad + \dots + CA^{N_p-N_c}B\Delta u(k_i + N_c - 1). & (1.11) \end{aligned}$$

Note that all predicted variables are formulated in terms of current state variable information $x(k_i)$ and the future control movement $\Delta u(k_i + j)$, where $j = 0, 1, \dots, N_c - 1$.

Define vectors

$$Y = [y(k_i + 1 \mid k_i) \ y(k_i + 2 \mid k_i) \ y(k_i + 3 \mid k_i) \ \dots \ y(k_i + N_p \mid k_i)]^T$$

$$\Delta U = [\Delta u(k_i) \ \Delta u(k_i + 1) \ \Delta u(k_i + 2) \ \dots \ \Delta u(k_i + N_c - 1)]^T,$$

where in the single-input and single-output case, the dimension of Y is N_p and the dimension of ΔU is N_c . We collect (1.10) and (1.11) together in a compact matrix form as

$$Y = Fx(k_i) + \Phi\Delta U, \quad (1.12)$$

where

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix}; \Phi = \begin{bmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ CA^2B & CAB & CB & \dots & 0 \\ \vdots & & & & \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \dots & CA^{N_p-N_c}B \end{bmatrix}.$$

1.3.2 Optimization

For a given set-point signal $r(k_i)$ at sample time k_i , within a prediction horizon the objective of the predictive control system is to bring the predicted output as close as possible to the set-point signal, where we assume that the set-point signal remains constant in the optimization window. This objective is then translated into a design to find the ‘best’ control parameter vector ΔU such that an error function between the set-point and the predicted output is minimized.

Assuming that the data vector that contains the set-point information is

$$R_s^T = \overbrace{[1 \ 1 \ \dots \ 1]}^{N_p} r(k_i),$$

we define the cost function J that reflects the control objective as

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \bar{R} \Delta U, \quad (1.13)$$

where the first term is linked to the objective of minimizing the errors between the predicted output and the set-point signal while the second term reflects the consideration given to the size of ΔU when the objective function J is made to be as small as possible. \bar{R} is a diagonal matrix in the form that $\bar{R} = r_w I_{N_c \times N_c}$ ($r_w \geq 0$) where r_w is used as a tuning parameter for the desired closed-loop performance. For the case that $r_w = 0$, the cost function (1.13) is interpreted as the situation where we would not want to pay any attention to how large the ΔU might be and our goal would be solely to make the error $(R_s - Y)^T (R_s - Y)$ as small as possible. For the case of large r_w , the cost function (1.13) is interpreted as the situation where we would carefully consider how large the ΔU might be and cautiously reduce the error $(R_s - Y)^T (R_s - Y)$.

To find the optimal ΔU that will minimize J , by using (1.12), J is expressed as

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U. \quad (1.14)$$

From the first derivative of the cost function J :

$$\frac{\partial J}{\partial \Delta U} = -2\Phi^T (R_s - Fx(k_i)) + 2(\Phi^T \Phi + \bar{R}) \Delta U, \quad (1.15)$$

the necessary condition of the minimum J is obtained as

$$\frac{\partial J}{\partial \Delta U} = 0,$$

from which we find the optimal solution for the control signal as

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i)), \quad (1.16)$$

with the assumption that $(\Phi^T \Phi + \bar{R})^{-1}$ exists. The matrix $(\Phi^T \Phi + \bar{R})^{-1}$ is called the Hessian matrix in the optimization literature. Note that R_s is a data vector that contains the set-point information expressed as

$$R_s = \underbrace{[1 \ 1 \ 1 \ \dots \ 1]^T}_{N_p} r(k_i) = \bar{R}_s r(k_i),$$

where

$$\bar{R}_s = \underbrace{[1 \ 1 \ 1 \ \dots \ 1]^T}_{N_p}.$$

The optimal solution of the control signal is linked to the set-point signal $r(k_i)$ and the state variable $x(k_i)$ via the following equation:

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (\bar{R}_s r(k_i) - Fx(k_i)). \quad (1.17)$$

Example 1.2. Suppose that a first-order system is described by the state equation:

$$\begin{aligned} x_m(k+1) &= ax_m(k) + bu(k) \\ y(k) &= x_m(k), \end{aligned} \quad (1.18)$$

where $a = 0.8$ and $b = 0.1$ are scalars. Find the augmented state-space model. Assuming a prediction horizon $N_p = 10$ and control horizon $N_c = 4$, calculate the components that form the prediction of future output Y , and the quantities $\Phi^T \Phi$, $\Phi^T F$ and $\Phi^T \bar{R}_s$. Assuming that at a time k_i ($k_i = 10$ for this example), $r(k_i) = 1$ and the state vector $x(k_i) = [0.1 \ 0.2]^T$, find the optimal solution ΔU with respect to the cases where $r_w = 0$ and $r_w = 10$, and compare the results.

Solution. The augmented state-space equation is

$$\begin{aligned} \begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} &= \begin{bmatrix} a & 0 \\ a & 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} b \\ b \end{bmatrix} \Delta u(k) \\ y(k) &= [0 \ 1] \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}. \end{aligned} \quad (1.19)$$

Based on (1.12), the F and Φ matrices take the following forms:

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ CA^4 \\ CA^5 \\ CA^6 \\ CA^7 \\ CA^8 \\ CA^9 \\ CA^{10} \end{bmatrix}; \Phi = \begin{bmatrix} CB & 0 & 0 & 0 \\ CAB & CB & 0 & 0 \\ CA^2B & CAB & CB & 0 \\ CA^3B & CA^2B & CAB & CB \\ CA^4B & CA^3B & CA^2B & CAB \\ CA^5B & CA^4B & CA^3B & CA^2B \\ CA^6B & CA^5B & CA^4B & CA^3B \\ CA^7B & CA^6B & CA^5B & CA^4B \\ CA^8B & CA^7B & CA^6B & CA^5B \\ CA^9B & CA^8B & CA^7B & CA^6B \end{bmatrix}.$$

The coefficients in the F and Φ matrices are calculated as follows:

$$\begin{aligned} CA &= [s_1 \ 1] \\ CA^2 &= [s_2 \ 1] \\ CA^3 &= [s_3 \ 1] \\ &\vdots \\ CA^k &= [s_k \ 1], \end{aligned} \quad (1.20)$$

where $s_1 = a$, $s_2 = a^2 + s_1$, ..., $s_k = a^k + s_{k-1}$, and

$$\begin{aligned} CB &= g_0 = b \\ CAB &= g_1 = ab + g_0 \\ CA^2B &= g_2 = a^2b + g_1 \\ &\vdots \\ CA^{k-1}B &= g_{k-1} = a^{k-1}b + g_{k-2} \\ CA^kB &= g_k = a^kb + g_{k-1}. \end{aligned} \quad (1.21)$$

With the plant parameters $a = 0.8$ and $b = 0.1$, $N_p = 10$ and $N_c = 4$, we calculate the quantities

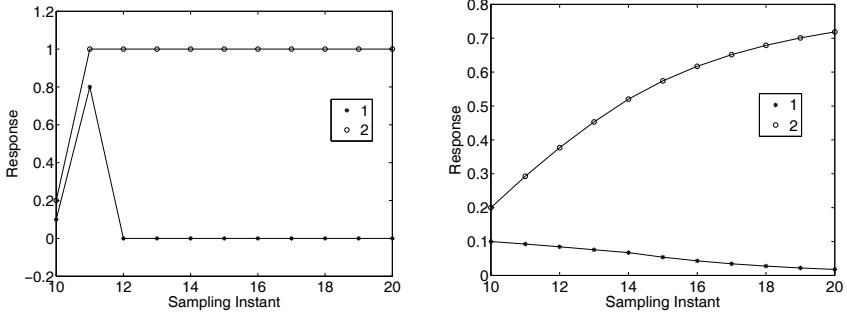
$$\begin{aligned} \Phi^T \Phi &= \begin{bmatrix} 1.1541 & 1.0407 & 0.9116 & 0.7726 \\ 1.0407 & 0.9549 & 0.8475 & 0.7259 \\ 0.9116 & 0.8475 & 0.7675 & 0.6674 \\ 0.7726 & 0.7259 & 0.6674 & 0.5943 \end{bmatrix} \\ \Phi^T F &= \begin{bmatrix} 9.2325 & 3.2147 \\ 8.3259 & 2.7684 \\ 7.2927 & 2.3355 \\ 6.1811 & 1.9194 \end{bmatrix}; \Phi^T \bar{R}_s &= \begin{bmatrix} 3.2147 \\ 2.7684 \\ 2.3355 \\ 1.9194 \end{bmatrix}. \end{aligned}$$

Note that the vector $\Phi^T \bar{R}_s$ is identical to the last column in the matrix $\Phi^T F$. This is because the last column of F matrix is identical to \bar{R}_s .

At time $k_i = 10$, the state vector $x(k_i) = [0.1 \ 0.2]^T$. In the first case, the error between predicted Y and R_s is reduced without any consideration to the magnitude of control changes. Namely, $r_w = 0$. Then, the optimal ΔU is found through the calculation

$$\Delta U = (\Phi^T \Phi)^{-1}(\Phi^T R_s - \Phi^T F x(k_i)) = [7.2 \ -6.4 \ 0 \ 0]^T.$$

We note that without weighting on the incremental control, the last two elements $\Delta u(k_i + 2) = 0$ and $\Delta u(k_i + 3) = 0$, while the first two elements have a rather large magnitude. Figure 1.1a shows the changes of the state variables where we can see that the predicted output y has reached the desired set-point



(a) State variables with no weight on Δu (b) State variables with weight on Δu

Fig. 1.1. Comparison of optimal solutions. Key: line (1) Δx_m ; line (2) y

1 while the Δx_m decays to zero. To examine the effect of the weight r_w on the optimal solution of the control, we let $r_w = 10$. The optimal solution of ΔU is given below, where I is a 4×4 identity matrix,

$$\begin{aligned}\Delta U &= (\Phi^T \Phi + 10 \times I)^{-1} (\Phi^T R_s - \Phi^T F x(k_i)) \\ &= [0.1269 \ 0.1034 \ 0.0829 \ 0.065]^T.\end{aligned}\quad (1.22)$$

With this choice, the magnitude of the first two control increments is significantly reduced, also the last two components are no longer zero. Figure 1.1b shows the optimal state variables. It is seen that the output y did not reach the set-point value of 1, however, the Δx_m approaches zero.

An observation follows from the comparison study. It seems that if we want the control to move cautiously, then it takes longer for the control signal to reach its steady state (*i.e.*, the values in ΔU decrease more slowly), because the optimal control energy is distributed over the longer period of future time. We can verify this by increasing N_c to 9, while maintaining $r_w = 10$. The result shows that the magnitude of the elements in ΔU is reducing, but they are significant for the first 8 elements:

$$\Delta U^T = [0.1227 \ 0.0993 \ 0.0790 \ 0.0614 \ 0.0463 \ 0.0334 \ 0.0227 \ 0.0139 \ 0.0072].$$

In comparison with the case where $N_c = 4$, we note that when $N_c = 9$, the first four parameters in ΔU are slightly different from the previous case.

Example 1.3. There is an alternative way to find the minimum of the cost function via completing the squares. This is an intuitive approach, also the minimum of the cost function becomes a by-product of the approach.

Find the optimal solution for ΔU by completing the squares of the cost function J (1.14).

Solution. From (1.14), by adding and subtracting the term

$$(R_s - Fx(k_i))^T \Phi (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i))$$

to the original cost function J , its value remains unchanged. This leads to

$$\begin{aligned} J = & (R_s - Fx(k_i))^T (R_s - Fx(k_i)) \\ & \underbrace{- 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U}_{\text{completed squares}} \\ & + \underbrace{(R_s - Fx(k_i))^T \Phi (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i))}_{(1.23)} \\ & - (R_s - Fx(k_i))^T \Phi (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i)), \end{aligned}$$

where the quantities under the \curvearrowleft are the completed ‘squares’:

$$\begin{aligned} J_0 = & (\Delta U - (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i)))^T \\ & \times (\Phi^T \Phi + \bar{R}) (\Delta U - (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i))). \end{aligned} \quad (1.24)$$

This can be easily verified by opening the squares. Since the first and last terms in (1.23) are independent of the variable ΔU (sometimes, we call this a decision variable), and $(\Phi^T \Phi + \bar{R})$ is assumed to be positive definite, then the minimum of the cost function J is achieved if the quantity J_0 equals zero, *i.e.*,

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i)). \quad (1.25)$$

This is the optimal control solution. By substituting this optimal solution into the cost function (1.23), we obtain the minimum of the cost as

$$\begin{aligned} J_{min} = & (R_s - Fx(k_i))^T (R_s - Fx(k_i)) \\ & - (R_s - Fx(k_i))^T \Phi (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - Fx(k_i)). \end{aligned}$$

1.3.3 MATLAB Tutorial: Computation of MPC Gains

Tutorial 1.2. *The objective of this tutorial is to produce a MATLAB function for calculating $\Phi^T \Phi$, $\Phi^T F$, $\Phi^T \bar{R}_s$. The key here is to create F and Φ matrices. Φ matrix is a Toeplitz matrix, which is created by defining its first column, and the next column is obtained through shifting the previous column.*

Step by Step

1. Create a new file called `mpcgain.m`.
2. The first step is to create the augmented model for MPC design. The input parameters to the function are the state-space model (A_p, B_p, C_p) , prediction horizon N_p and control horizon N_c . Enter the following program into the file:

```

function [Phi_Phi,Phi_F,Phi_R,A_e, B_e,C_e]
    =mpcgain(Ap,Bp,Cp,Nc,Np);
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
A_e=eye(n1+m1,n1+m1);
A_e(1:n1,1:n1)=Ap;
A_e(n1+1:n1+m1,1:n1)=Cp*Ap;
B_e=zeros(n1+m1,n_in);
B_e(1:n1,:)=Bp;
B_e(n1+1:n1+m1,:)=Cp*Bp;
C_e=zeros(m1,n1+m1);
C_e(:,n1+1:n1+m1)=eye(m1,m1);

```

3. Note that the F and Φ matrices have special forms. By taking advantage of the special structure, we obtain the matrices.
4. Continue entering the program into the file:

```

n=n1+m1;
h(1,:)=C_e;
F(1,:)=C_e*A_e;
for kk=2:Np
    h(kk,:)=h(kk-1,:)*A_e;
    F(kk,:)= F(kk-1,:)*A_e;
end
v=h*B_e;
Phi=zeros(Np,Nc); %declare the dimension of Phi
Phi(:,1)=v; % first column of Phi
for i=2:Nc
    Phi(:,i)=[zeros(i-1,1);v(1:Np-i+1,1)]; %Toeplitz matrix
end
BarRs=ones(Np,1);
Phi_Phi= Phi'*Phi;
Phi_F= Phi'*F;
Phi_R=Phi'*BarRs;

```

5. Type into the MATLAB Work Space with $Ap = 0.8$, $Bp = 0.1$, $Cp = 1$, $Nc = 4$ and $Np = 10$. Run this MATLAB function by typing

```
[Phi_Phi,Phi_F,Phi_R,A_e, B_e,C_e]
=mpcgain(Ap,Bp,Cp,Nc,Np);
```

6. Comparing the results with the answers from Example 1.2. If it is identical to what was presented there, then your program is correct.
7. Varying the prediction horizon and control horizon, observe the changes in these matrices.
8. Calculate ΔU by assuming the information of initial condition on x and r . The inverse of matrix M is calculated in MATLAB as $inv(M)$.
9. Validate the results in Example 1.2.

1.4 Receding Horizon Control

Although the optimal parameter vector ΔU contains the controls $\Delta u(k_i)$, $\Delta u(k_i + 1)$, $\Delta u(k_i + 2)$, ..., $\Delta u(k_i + N_c - 1)$, with the receding horizon control principle, we only implement the first sample of this sequence, *i.e.*, $\Delta u(k_i)$, while ignoring the rest of the sequence. When the next sample period arrives, the more recent measurement is taken to form the state vector $x(k_i + 1)$ for calculation of the new sequence of control signal. This procedure is repeated in real time to give the receding horizon control law.

Example 1.4. We illustrate this procedure by continuing Example 1.2, where a first-order system with the state-space description

$$x_m(k+1) = 0.8x_m(k) + 0.1u(k)$$

is used in the computation. We will consider the case $r_w = 0$. The initial conditions are $x(10) = [0.1 \ 0.2]^T$ and $u(9) = 0$.

Solution. At sample time $k_i = 10$, the optimal control was previously computed as $\Delta u(10) = 7.2$. Assuming that $u(9) = 0$, then the control signal to the plant is $u(10) = u(9) + \Delta u(10) = 7.2$ and with $x_m(10) = y(10) = 0.2$, we calculate the next simulated plant state variable

$$x_m(11) = 0.8x_m(10) + 0.1u(10) = 0.88. \quad (1.26)$$

At $k_i = 11$, the new plant information is $\Delta x_m(11) = 0.88 - 0.2 = 0.68$ and $y(11) = 0.88$, which forms $x(11) = [0.68 \ 0.88]^T$. Then we obtain

$$\Delta U = (\Phi^T \Phi)^{-1} (\Phi^T R_s - \Phi^T F x(11)) = [-4.24 \ -0.96 \ 0.0000 \ 0.0000]^T.$$

This leads to the optimal control $u(11) = u(10) + \Delta u(11) = 2.96$. This new control is implemented to obtain

$$x_m(12) = 0.8x_m(11) + 0.1u(11) = 1. \quad (1.27)$$

At $k_i = 12$, the new plant information is $\Delta x_m(12) = 1 - 0.88 = 0.12$ and $y(12) = 1$, which forms $x(12) = [0.12 \ 1]^T$. We obtain

$$\Delta U = (\Phi^T \Phi)^{-1} (\Phi^T R_s - \Phi^T F x(11)) = [-0.96 \ 0.000 \ 0.0000 \ 0.0000]^T.$$

This leads to the control at $k_i = 12$ as $u(12) = u(11) - 0.96 = 2$. By implementing this control, we obtain the next plant output as

$$x_m(13) = ax_m(12) + bu(12) = 1. \quad (1.28)$$

The new plant information is $\Delta x_m(13) = 1 - 1 = 0$ and $y(13) = 1$. From this information, we obtain

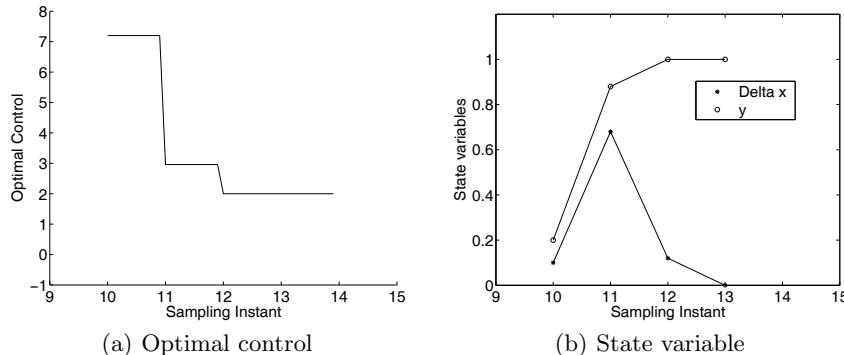


Fig. 1.2. Receding horizon control

$$\Delta U = (\Phi^T \Phi)^{-1} (\Phi^T R_s - \Phi^T F x(11)) = \begin{bmatrix} 0.000 & 0.000 & 0.0000 & 0.0000 \end{bmatrix}^T.$$

Figure 1.2 shows the trajectories of the state variable Δx_m and y , as well as the control signal that was used to regulate the output. This example also illustrated the differences between the ΔU parameter vectors at different time instances. We note that as the output response reaches the desired set-point signal, the parameters in the ΔU approach zero.

1.4.1 Closed-loop Control System

There is another aspect that was illustrated by Example 1.4. If we examine this example carefully, then we find that at a given time k_i , the optimal parameter vector ΔU is solved using

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T R_s - \Phi^T F x(k_i)),$$

where $(\Phi^T \Phi + \bar{R})^{-1} \Phi^T R_s$ corresponds to the set-point change, while $-(\Phi^T \Phi + \bar{R})^{-1} \Phi^T F$ corresponds to the state feedback control within the framework of predictive control. Both depend on the system parameters, hence are constant matrices for a time-invariant system. Because of the receding horizon control principle, we only take the first element of ΔU at time k_i as the incremental control, thus

$$\begin{aligned}\Delta u(k_i) &= \overbrace{\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}}^{N_c} (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T \bar{R}_s r(k_i) - \Phi^T F x(k_i)) \\ &= K_y r(k_i) - K_{mpc} x(k_i),\end{aligned}\quad (1.29)$$

where K_y is the first element of

$$(\Phi^T \Phi + \bar{R})^{-1} \Phi^T \bar{R}_s.$$

and K_{mpc} is the first row of

$$(\Phi^T \Phi + \bar{R})^{-1} \Phi^T F.$$

Equation (1.29) is in a standard form of linear time-invariant state feedback control. The state feedback control gain vector is K_{mpc} . Therefore, with the augmented design model

$$x(k+1) = Ax(k) + B\Delta u(k)$$

the closed-loop system is obtained by substituting (1.29) into the augmented system equation; changing index k_i to k , leading to the closed-loop equation

$$x(k+1) = Ax(k) - BK_{mpc}x(k) + BK_y r(k) \quad (1.30)$$

$$= (A - BK_{mpc})x(k) + BK_y r(k). \quad (1.31)$$

Thus, the closed-loop eigenvalues can be evaluated through the closed-loop characteristic equation:

$$\det[\lambda I - (A - BK_{mpc})] = 0.$$

Because of the special structures of the matrices C and A , the last column of F is identical to \bar{R}_s , which is $[1 \ 1 \ \dots, \ 1]^T$, therefore K_y is identical to the last element of K_{mpc} . Noting that the state variable vector $x(k_i) = [\Delta x_m(k)^T \ y(k)]^T$, and with the definition of K_y , we can write $K_{mpc} = [K_x \ K_y]$, where K_x corresponds to the feedback gain vector related to $\Delta x_m(k)$, and K_y corresponds to the feedback gain related to $y(k)$. Then, we obtain the closed-loop block diagram for the predictive control system as in Figure 1.3 where q^{-1} denotes the backward shift operator. The diagram shows the state feedback structure for the discrete model prediction control (DMPC) with integral action in which the module $\frac{1}{1-q^{-1}}$ denotes the discrete-time integrator.

Example 1.5. This example will examine the closed-loop feedback gain matrices generated from Example 1.2 and the eigenvalues of the closed-loop system with weight $r_w = 0$ and $r_w = 10$.

Solution. When the weight $r_w = 0$, we have

$$K_y = [1 \ 0 \ 0 \ 0] (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}) = 10$$

$$K_{mpc} = [1 \ 0 \ 0 \ 0] (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T F) = [8 \ 10].$$

Hence, the eigenvalues of the closed-loop system are calculated by evaluating the eigenvalues of the closed-loop matrix $A - BK_{mpc}$, where, from Example 1.2

$$A = \begin{bmatrix} 0.8 & 0 \\ 0.8 & 1 \end{bmatrix}; \quad B = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}.$$

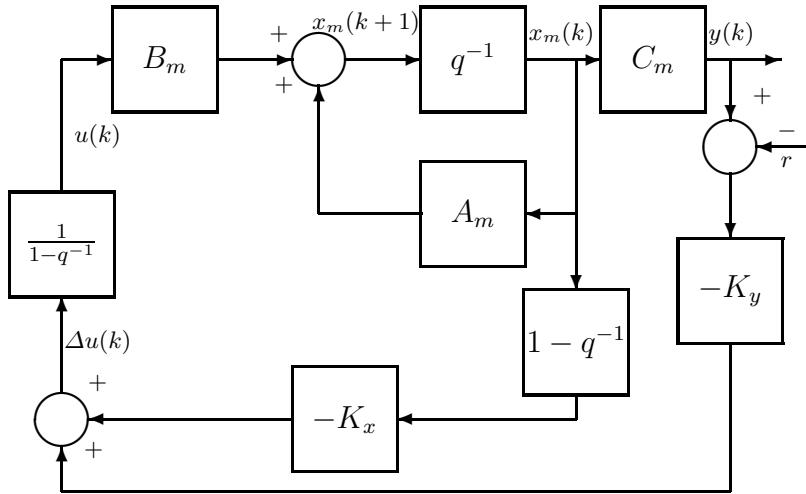


Fig. 1.3. Block diagram of discrete-time predictive control system

They are $\lambda_1 = -6.409 \times 10^{-7}$ and $\lambda_2 = 6.409 \times 10^{-7}$, approximately on the origin of the complex plane.

When the weight $r_w = 10$, we have

$$K_y = [1 \ 0 \ 0 \ 0] (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}) = 0.2453$$

$$K_{mpc} = [1 \ 0 \ 0 \ 0] (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T F) = [0.6939 \ 0.2453].$$

With this gain vector, the eigenvalues of the closed-loop system are $\lambda_{1,2} = 0.8530 \pm j0.0542$, indicating that the dynamics of the closed-loop system have a much slower response than the one in the case when $r_w = 0$.

Example 1.6. Suppose that a continuous-time system is described by the Laplace transfer function

$$G(s) = \frac{\omega^2}{s^2 + 0.1\omega s + \omega^2},$$

where $\omega = 10$. This system is discretized using a sampling interval $\Delta t = 0.01$. Examine sensitivity issues in the selection of design parameters with $N_c = 3$, $N_p = 20$ and 200 ; $\bar{R} = 0.5I$.

Solution. We first use the MATLAB function script, given below, to obtain the continuous-time state-space model, then by following Tutorial 1.1 obtain the discrete-time state-space model.

```

omega=10;
numc=omega^2;
denc=[1 0.1*omega omega^2];
[Ac,Bc,Cc,Dc]=tf2ss(numc,denc);

```

Here, the augmented discrete-time state-space equation is

$$\begin{aligned}x(k+1) &= Ax(k) + B\Delta u(k) \\y(k) &= Cx(k),\end{aligned}$$

where

$$A = \begin{bmatrix} 0.9851 & -0.9934 & 0 \\ 0.0099 & 0.9950 & 0 \\ 0.9934 & 99.5021 & 1 \end{bmatrix}; B = \begin{bmatrix} 0.0099 \\ 0.0000 \\ 0.0050 \end{bmatrix}$$

$$C = [0 \ 0 \ 1].$$

Let us first look at the effect of the prediction horizon on the solution of ΔU . We assume that at sampling instant $k = 10$, the initial condition of $x(10) = [0.1 \ 0.2 \ 0.3]^T$. The solution of ΔU for $N_p = 20$ is $\Delta U = [-144.9984 \ -65.4710 \ 1.2037]^T$. By using the receding horizon control principle, the state feedback control gain is $K_{mpc} = [45.4168 \ 705.6132 \ 0.9513]$, and the closed-loop eigenvalues are $0.6974, 0.8959 \pm 0.1429j$. However, the solution of ΔU for $N_p = 200$ is $\Delta U = [-645.5885 \ -0.4664 \ 629.0276]^T$. In comparison with the previous case where the shorter prediction horizon $N_p = 20$ was used, the parameter vector ΔU has changed significantly. Again, from using the receding horizon control principle, the state feedback control gain is $K_{mpc} = [80.6 \ 3190 \ 0.79]$ and the resultant closed-loop eigenvalues are $0.9749, 0.5207 \pm j0.2919$. This comparison study illustrated the existing sensitivity in the design with respect to the choice of prediction horizon. Looking closely, we will discover that the Hessian matrix

$$\Phi^T \Phi + \bar{R}$$

is a function of the prediction horizon. For example, for $N_p = 20$

$$\Phi^T \Phi = \begin{bmatrix} 9.8796 & 8.9387 & 8.0099 \\ 8.9387 & 8.1020 & 7.2737 \\ 8.0099 & 7.2737 & 6.5425 \end{bmatrix},$$

with condition number $\kappa(\Phi^T \Phi + 0.5I) = 49.98$. However, for $N_p = 200$,

$$\Phi^T \Phi = \begin{bmatrix} 236.0557 & 235.5010 & 234.5466 \\ 235.5010 & 235.3753 & 234.8473 \\ 234.5466 & 234.8473 & 234.7473 \end{bmatrix},$$

with condition number $\kappa(\Phi^T \Phi + 0.5I) = 1410$. The condition number of the Hessian matrix has significantly increased as the prediction horizon N_p increased to 200. This large condition number of the Hessian matrix for a long

prediction horizon results in the numerical sensitivity that causes the significant difference between the short and long prediction horizon cases.

With short prediction and control horizons, the closed-loop predictive control system is not necessarily stable. Traditionally, these are the tuning parameters for closed-loop stability and performance. In Chapter 4, we will propose an approach that uses large prediction and control horizons so as to guarantee closed-loop stability.

1.4.2 MATLAB Tutorial: Implementation of Receding Horizon Control

Tutorial 1.3. *The objective of this tutorial is to learn how to implement a predictive control system using receding horizon control. The plant state-space model is given by*

$$\begin{aligned}x_m(k+1) &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_m(k) + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u(k) \\y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x_m(k).\end{aligned}\quad (1.32)$$

Step by Step

1. Create a new file called *reced.m*
2. The first step is to define the plant, enter the values of prediction horizon and control horizon. The plant is the double integrator system (1.32). Control horizon is selected to be $N_c = 4$ and prediction horizon is $N_p = 20$. Enter the following program into the file:

```
Ap=[1 1;0 1];
Bp=[0.5;1];
Cp=[1 0];
Dp=0;
Np=20;
Nc=4;
```

3. The program calls the function *mpcgain.m* to generate the necessary gain matrices and specifies the initial conditions for implementation of receding horizon control. The initial state variable for the plant is $xm=0$; and the initial state feedback variable is $Xf=0$; set-point signal is specified and the number of simulation points is specified as 100.
4. Continue entering the following program into the file:

```
[Phi_Phi,Phi_F,Phi_R,A_e, B_e,C_e]
    = mpchgain(Ap,Bp,Cp,Nc,Np);
[n,n_in]=size(B_e);
xm=[0;0];
Xf=zeros(n,1);
N_sim=100;
```

```
r=ones(N_sim,1);
u=0; % u(k-1) =0
y=0;
```

5. From the receding horizon control, at sample time kk , the ΔU vector is calculated using the set-point signal $r(kk)$ and the state vector Xf . Then, $\Delta u(kk)$ is taken as the first element of ΔU ; and $u(kk) = u(kk-1) + \Delta u(k)$. The weight factor is selected as 0.1.

6. Continue entering the following program into the file:

```
for kk=1:N_sim;
DeltaU=inv(Phi_Phi+0.1*eye(Nc,Nc))*(Phi_R*r(kk)-Phi_F*Xf);
deltau=DeltaU(1,1);
u=u+deltau;
u1(kk)=u;
y1(kk)=y;
```

7. The plant state and output are simulated using the control signal generated; the state variable used in the feedback mechanism is updated as Xf .

8. Continue entering the following program into the file:

```
xm_old=xm;
xm=Ap*xm+Bp*u;
y=Cp*xm;
Xf=[xm-xm_old;y];
end
```

9. The input and output signals are plotted against samples.

10. Continue entering the following program into the file:

```
k=0:(N_sim-1);
figure
subplot(211)
plot(k,y1)
xlabel('Sampling Instant')
legend('Output')
subplot(212)
plot(k,u1)
xlabel('Sampling Instant')
legend('Control')
```

11. Save the program in the same directory as the one that contains the function. Run the program.
 12. Change the weight r_w in the design to 2 and observe that the closed-loop response speed is slower.
 13. Create your own plant using different Ap , Bp and Cp , and experiment with different prediction and control horizons.

1.5 Predictive Control of MIMO Systems

In the previous section, for simplicity of illustration the predictive control system was designed based on a single-input and single-output system. This design methodology can be readily extended to multi-input and multi-output systems without much additional effort, because of the state-space formulation.

1.5.1 General Formulation of the Model

Assume that the plant has m inputs, q outputs and n_1 states. We also assume that the number of outputs is less than or equal to the number of inputs (*i.e.*, $q \leq m$). If the number of outputs is greater than the number of inputs, we cannot hope to control each of the measured outputs independently with zero steady-state errors. In the general formulation of the predictive control problem, we also take the plant noise and disturbance into consideration.

$$x_m(k+1) = A_m x_m(k) + B_m u(k) + B_d \omega(k) \quad (1.33)$$

$$y(k) = C_m x_m(k), \quad (1.34)$$

where $\omega(k)$ is the input disturbance, assumed to be a sequence of integrated white noise. This means that the input disturbance $\omega(k)$ is related to a zero-mean, white noise sequence $\epsilon(k)$ by the difference equation

$$\omega(k) - \omega(k-1) = \epsilon(k). \quad (1.35)$$

Note that from (1.33), the following difference equation is also true:

$$x_m(k) = A_m x_m(k-1) + B_m u(k-1) + B_d \omega(k-1). \quad (1.36)$$

By defining $\Delta x_m(k) = x_m(k) - x_m(k-1)$ and $\Delta u(k) = u(k) - u(k-1)$, then subtracting (1.36) from (1.33) leads to

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u(k) + B_d \epsilon(k). \quad (1.37)$$

In order to relate the output $y(k)$ to the state variable $\Delta x_m(k)$, we deduce that

$$\Delta y(k+1) = C_m \Delta x_m(k+1) = C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k) + C_m B_d \epsilon(k),$$

where $\Delta y(k+1) = y(k+1) - y(k)$.

Choosing a new state variable vector $x(k) = [\Delta x_m(k)^T \ y(k)^T]^T$, we have:

$$\begin{aligned} \begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} &= \begin{bmatrix} A_m & o_m^T \\ C_m A_m & I_{q \times q} \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix} \Delta u(k) \\ &\quad + \begin{bmatrix} B_d \\ C_m B_d \end{bmatrix} \epsilon(k) \\ y(k) &= [o_m \ I_{q \times q}] \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}, \end{aligned} \quad (1.38)$$

where $I_{q \times q}$ is the identity matrix with dimensions $q \times q$, which is the number of outputs; and o_m is a $q \times n_1$ zero matrix. In (1.38), A_m , B_m and C_m have dimension $n_1 \times n_1$, $n_1 \times m$ and $q \times n_1$, respectively.

For notational simplicity, we denote (1.38) by

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) + B_\epsilon\epsilon(k) \\ y(k) &= Cx(k), \end{aligned} \quad (1.39)$$

where A , B and C are matrices corresponding to the forms given in (1.38). In the following, the dimensionality of the augmented state-space equation is taken to be n ($= n_1 + q$).

There are two points that are worth investigating here. The first is related to the eigenvalues of the augmented design model. The second point is related to the realization of the state-space model. Both points will help us understand the model.

Eigenvalues of the Augmented Model

Note that the characteristic polynomial equation of the augmented model is

$$\begin{aligned} \rho(\lambda) &= \det \begin{bmatrix} \lambda I - A_m & o_m^T \\ -C_m A_m & (\lambda - 1)I_{q \times q} \end{bmatrix} \\ &= (\lambda - 1)^q \det(\lambda I - A_m) = 0 \end{aligned} \quad (1.40)$$

where we used the property that the determinant of a block lower triangular matrix equals the product of the determinants of the matrices on the diagonal. Hence, the eigenvalues of the augmented model are the union of the eigenvalues of the plant model and the q eigenvalues, $\lambda = 1$. This means that there are q integrators embedded into the augmented design model. This is the means we use to obtain integral action for the MPC systems.

Controllability and Observability of the Augmented Model

Because the original plant model is augmented with integrators and the MPC design is performed on the basis of the augmented state-space model, it is important for control system design that the augmented model does not become uncontrollable or unobservable, particularly with respect to the unstable dynamics of the system. Controllability is a pre-requisite for the predictive control system to achieve the desired closed-loop control performance and observability is a pre-requisite for a successful design of an observer. However, the conditions may be relaxed to the requirement of stabilizability and detectability, if only closed-loop stability is of concern.¹ In this book, unless it is

¹ A system is stabilizable if its uncontrollable modes, if any, are stable. Its controllable modes may be stable or unstable. A system is detectable, if its unobservable modes, if any, are stable. Its observable modes may be stable or unstable. Stable modes here means that the corresponding eigenvalues are strictly inside the unit circle.

specifically stated, we require the model to be both controllable and observable in order to achieve desired closed-loop performance. An example is given in Section 1.6 to illustrate the importance of observability for the design of observer.

Because the augmented model introduced additional integral modes, we need to examine under what conditions these additional modes become controllable. The simplest way for the investigation is based on the assumption of minimal realization of the plant model. The discussion of minimal realization, controllability and observability can be found in control textbooks (for example Kailath (1980), Bay (1999)).

Definition: A realization of transfer function $G(z)$ is any state-space triplet (A, B, C) such that $G(z) = C(zI - A)^{-1}B$. If such a set (A, B, C) exists, then $G(z)$ is said to be realizable. A realization (A, B, C) is called a minimal realization of a transfer function if no other realization of smaller dimension of the triplet exists.

A minimal realization has the distinctive feature summarized in the theorem below.

Theorem 1.1. *A minimal realization is both controllable and observable (Kailath, 1980, Bay, 1999).*

With this background information, we aim to show conditions such that the augmented model is both controllable and observable through the argument of minimal realization.

Theorem 1.2. *Assume that the plant model (A_m, B_m, C_m) is both controllable and observable having the transfer function $G_m(z)$ with minimal realization, where*

$$G_m(z) = C_m(zI - A_m)^{-1}B_m.$$

Then, the transfer function of augmented design model (1.39) has the representation

$$G(z) = \frac{z}{z-1}G_m(z), \quad (1.41)$$

*and is both controllable and observable if and only if the plant model $G_m(z)$ has no zero at $z = 1$.*²

Proof. To prove that the augmented model is controllable and observable, we need to show that (1.41) is true. After that, the results follow from the minimal structure of the augmented model without pole-zero cancellation.

Note that for a given square matrix M with the block structure

$$M = \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix},$$

² The zeros of a MIMO transfer function are those values of z that make the matrix $G_m(z)$ lose rank.

if A_{11}^{-1} and A_{22}^{-1} exist, then

$$M^{-1} = \begin{bmatrix} A_{11}^{-1} & 0 \\ -A_{22}^{-1}A_{21}A_{11}^{-1} & A_{22}^{-1} \end{bmatrix}. \quad (1.42)$$

By applying the equality (1.42), we obtain

$$G(z) = C(zI - A)^{-1}B, \quad (1.43)$$

where

$$(zI - A)^{-1} = \begin{bmatrix} (zI_m - A_m)^{-1} & 0 \\ (1 - z^{-1})C_mA_m(zI_m - A_m)^{-1} & (1 - z^{-1})I_q \end{bmatrix}.$$

By substituting the B and C matrices from (1.39), the transfer function of the augmented model is obtained as (1.41). Under the assumption that the plant model $G_m(z)$ has no zero at $z = 1$ and has a minimal realization, the transfer function of the augmented model has a minimal structure from (1.41), therefore it is both controllable and observable.

For a single-input, single-output system, if one of the zeros of the transfer function is at $z = 1$, then the augmented model is not controllable. For instance, if

$$G_m(z) = \frac{(z - 1)}{(z - 0.6)(z - 0.8)},$$

then there will be a pole-zero cancellation in $G(z)$, which is

$$G(z) = \frac{z}{z - 1} \frac{(z - 1)}{(z - 0.6)(z - 0.8)}.$$

In the single-input, single-output case, the steady-state gain of the plant model is zero, and it does not permit integral control.

We emphasize that the number of inputs is greater than or equal to the number of outputs ($m \geq q$). When the number of inputs is less than the number of outputs, the augmented integral modes may become uncontrollable.

When using MATLAB, minimal realization of a state-space model is achieved through model-order reduction. For example, when a discrete-time transfer function is

$$G_m(z) = \frac{(z - 0.1)}{(z - 0.1)(z - 0.9)},$$

there is a pole-zero cancellation at $z = 0.1$. One of the realizations of the state-space model based on $G_m(z)$ using MATLAB function (tf2ss.m) has two state variables with

$$A_m = \begin{bmatrix} 1 & -0.09 \\ 1 & 0 \end{bmatrix}; B_m = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; C_m = [1 \ 0].$$

This is not a minimal realization as the corresponding transfer function has a pole-zero cancellation. To obtain a minimal state-space realization, the following MATLAB script is used for this simple illustrative example.

```

numd=[1 -0.1];
dend=conv([1 -0.1],[1 -0.9]);
sys1=tf(numd,dend);
sys=ss(sys1,'min');
[Am,Bm,Cm,Dm]=ssdata(sys);

```

The minimal realization through model-order reduction is

$$A_m = 0.9; \quad B_m = -0.9806; \quad C_m = -1.0198,$$

which only has one state variable as we expected in a minimal realization for this example.

1.5.2 Solution of Predictive Control for MIMO Systems

The extension of the predictive control solution is quite straightforward, and we need to pay attention to the dimensions of the state, control and output vectors in a multi-input, multi-output environment. Define the vectors Y and ΔU as

$$\Delta U = [\Delta u(k_i)^T \quad \Delta u(k_i + 1)^T \quad \dots \quad \Delta u(k_i + N_c - 1)^T]^T$$

$$Y = [y(k_i + 1 | k_i)^T \quad y(k_i + 2 | k_i)^T \quad y(k_i + 3 | k_i)^T \quad \dots \quad y(k_i + N_p | k_i)^T]^T.$$

Based on the state-space model (A, B, C) , the future state variables are calculated sequentially using the set of future control parameters

$$\begin{aligned}
x(k_i + 1 | k_i) &= Ax(k_i) + B\Delta u(k_i) + B_d\epsilon(k_i) \\
x(k_i + 2 | k_i) &= Ax(k_i + 1 | k_i) + B\Delta u(k_i + 1) + B_d\epsilon(k_i + 1 | k_i) \\
&= A^2x(k_i) + AB\Delta u(k_i) + B\Delta u(k_i + 1) \\
&\quad + AB_d\epsilon(k_i) + B_d\epsilon(k_i + 1 | k_i) \\
&\vdots \\
x(k_i + N_p | k_i) &= A^{N_p}x(k_i) + A^{N_p-1}B\Delta u(k_i) + A^{N_p-2}B\Delta u(k_i + 1) \\
&\quad + A^{N_p-N_c}B\Delta u(k_i + N_c - 1) + A^{N_p-1}B_d\epsilon(k_i) \\
&\quad + A^{N_p-2}B_d\epsilon(k_i + 1 | k_i) + \dots + B_d\epsilon(k_i + N_p - 1 | k_i).
\end{aligned}$$

With the assumption that $\epsilon(k)$ is a zero-mean white noise sequence, the predicted value of $\epsilon(k_i + i | k_i)$ at future sample i is assumed to be zero. The prediction of state variable and output variable is calculated as the expected values of the respective variables, hence, the noise effect to the predicted values being zero. For notational simplicity, the expectation operator is omitted without confusion.

Effectively, we have

$$Y = Fx(k_i) + \Phi\Delta U, \quad (1.44)$$

where

$$F = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix}; \Phi = \begin{bmatrix} CB & 0 & 0 & \dots & 0 \\ CAB & CB & 0 & \dots & 0 \\ CA^2B & CAB & CB & \dots & 0 \\ \vdots & \vdots & & & \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \dots & CA^{N_p-N_c}B \end{bmatrix}.$$

The incremental optimal control within one optimization window is given by

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T \bar{R}_s r(k_i) - \Phi^T F x(k_i)), \quad (1.45)$$

where matrix $\Phi^T \Phi$ has dimension $m N_c \times m N_c$ and $\Phi^T F$ has dimension $m N_c \times n$, and $\Phi^T \bar{R}_s$ equals the last q columns of $\Phi^T F$. The weight matrix \bar{R} is a block matrix with m blocks and has its dimension equal to the dimension of $\Phi^T \Phi$. The set-point signal is $r(k_i) = [r_1(k_i) \ r_2(k_i) \ \dots \ r_q(k_i)]^T$ as the q set-point signals to the multi-output system.

Applying the receding horizon control principle, the first m elements in ΔU are taken to form the incremental optimal control:

$$\begin{aligned} \Delta u(k_i) &= \underbrace{[I_m \ o_m \ \dots \ o_m]}_{N_c} (\Phi^T \Phi + \bar{R})^{-1} (\Phi^T \bar{R}_s r(k_i) - \Phi^T F x(k_i)) \\ &= K_y r(k_i) - K_{mpc} x(k_i), \end{aligned} \quad (1.46)$$

where I_m and o_m are, respectively, the identity and zero matrix with dimension $m \times m$.

Further work on predictive control approaches to MIMO systems will be presented in Chapter 3, where Laguerre functions will be used in the design. In that chapter, MATLAB functions will be given for the design of MIMO predictive control systems.

1.6 State Estimation

In the design of model predictive controllers, we assumed that the information $x(k_i)$ is available at the time k_i . This assumes that all the state variables are measurable. In reality, with most applications, not all state variables are measured (or available). Some of them may be impossible to measure. One approach is to select the state variables corresponding to the input and output using a special state-space realization (see Chapter 9) and the alternative is to estimate the state variable $x(k)$ from the process measurement. The ‘soft’ instrument used to estimate unknown state variables based on process measurement, in a control engineering context, is called an observer. The concept of an observer has been widely used in the science and engineering fields. In addition, in a noisy environment, a state observer can also act like a noise filter to reduce the effect of noise on the measurement. Our focus here is to use an observer in the design of predictive control.

1.6.1 Basic Ideas About an Observer

It would not be difficult to imagine that an observer is constructed based on a mathematical model of the plant. For instance, we assume the plant state-space model:

$$x_m(k+1) = A_m x_m(k) + B_m u(k). \quad (1.47)$$

Then we can use this model to calculate the state variable $\hat{x}_m(k)$, $k = 1, 2, \dots$, with an initial state condition $\hat{x}_m(0)$ and input signal $u(k)$ as

$$\hat{x}_m(k+1) = A_m \hat{x}_m(k) + B_m u(k). \quad (1.48)$$

This approach, in fact, would work after some transient time, if the plant model is stable and our guess of the initial condition is nearly correct. What could be the problems with this type of approach? Basically, it is an open-loop prediction. The error $\tilde{x}_m(k) = x_m(k) - \hat{x}_m(k)$ satisfies the difference equation:

$$\begin{aligned} \tilde{x}_m(k+1) &= A_m(x_m(k) - \hat{x}_m(k)) \\ &= A_m \tilde{x}_m(k). \end{aligned} \quad (1.49)$$

For a given initial error state $\tilde{x}_m(0) \neq 0$, we have

$$\tilde{x}_m(k) = A_m^k \tilde{x}_m(0). \quad (1.50)$$

Two points are discussed here. If A_m has all eigenvalues inside the unit circle, then the error system (1.50) is stable and $\|\tilde{x}_m(k)\| \rightarrow 0$ as $k \rightarrow \infty$, which means that the estimated state variable $\hat{x}_m(k)$ converges to $x_m(k)$. However, if A_m has one or more eigenvalues outside the unit circle, then the error system (1.50) is unstable and $\|\tilde{x}_m(k)\| \rightarrow \infty$ as $k \rightarrow \infty$, which means that the prediction $\hat{x}_m(k)$ does not converge to $x_m(k)$. If A_m has one or more eigenvalues on the unit circle, the error states $\|\tilde{x}_m(k)\|$ will not converge to zero. The second point is that in the case of a stable plant model A_m , we have no ‘control’ on the convergence rate of the error $\|\tilde{x}_m(k)\| \rightarrow 0$, which is dependent on the location of the plant poles. Namely, if the plant poles are close to the origin of the complex plane, then the error converges at a fast rate to zero; otherwise, the convergence rate could be slow.

The question is how to improve the estimate of $x_m(k)$. The solution is to use a feedback principle where an error signal is deployed to improve the estimation. The observer is constructed using the equation:

$$\hat{x}_m(k+1) = \overbrace{A_m \hat{x}_m(k) + B_m u(k)}^{\text{model}} + \overbrace{K_{ob}(y(k) - C_m \hat{x}_m(k))}^{\text{correction term}}, \quad (1.51)$$

where K_{ob} is the observer gain matrix. In the observer form, the state variable estimate $\hat{x}_m(k+1)$ consists of two terms. The first term is the original model,

and the second term is the correction term based on the error between the measured output and the predicted output using the estimate $\hat{x}_m(k)$.

To choose the observer gain K_{ob} , we examine the closed-loop error equation. By substituting $y(k) = C_m x_m(k)$ into (1.51), with the definition of error state $\tilde{x}_m(k) = x_m(k) - \hat{x}_m(k)$, we obtain that

$$\begin{aligned}\tilde{x}_m(k+1) &= A_m \tilde{x}_m(k) - K_{ob} C_m \tilde{x}_m(k) \\ &= (A_m - K_{ob} C_m) \tilde{x}_m(k).\end{aligned}\quad (1.52)$$

Now, with given initial error $\tilde{x}_m(0)$, we have

$$\tilde{x}_m(k) = (A_m - K_{ob} C_m)^k \tilde{x}_m(0). \quad (1.53)$$

Comparing the observer error response given by (1.53) with the open-loop prediction (1.50), it is apparent that the observer gain K_{ob} can be used to manipulate the convergence rate of the error. If there is only a single output, a commonly used approach is to place the closed-loop eigenvalues of the error system matrix $A_m - K_{ob} C_m$ at a desired location of the complex plane. The following example shows how to select the observer gain K_{ob} .

Example 1.7. The linearized equation of motion of a simple pendulum is

$$\frac{d^2\theta}{dt^2} + \omega^2\theta = u, \quad (1.54)$$

where θ is the angle of the pendulum. Design an observer that reconstructs the angle of the pendulum given measurements of $\frac{d\theta}{dt}$. Assume $\omega = 2$ rad/sec and sampling interval $\Delta t = 0.1$ (sec). The desired observer poles are chosen to be 0.1 and 0.2. Compare the open-loop estimation with the observer-based estimation.

Solution. Let $x_1(t) = \theta$ and $x_2(t) = \dot{\theta}$, using the motion equation (1.54), the corresponding state-space model is

$$\begin{aligned}\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= [0 \ 1] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}.\end{aligned}\quad (1.55)$$

With $\omega = 2$ rad/sec and sampling interval $\Delta t = 0.1$ (sec), the corresponding discrete-time state-space model is

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9801 & 0.0993 \\ -0.3973 & 0.9801 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.0050 \\ 0.09930 \end{bmatrix} u(k) \quad (1.56)$$

$$y(k) = [0 \ 1] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}. \quad (1.57)$$

To begin this study, we investigate what happens if the pendulum model alone is used for the prediction of the angle $\theta(x_1)$. Assume that the input signal $u(k) = 0$ and the initial conditions of the state variables are $\theta(0) = x_1(0) = 1$ and $\dot{\theta}(0) = x_2(0) = 0$. The trajectories of movement for both θ and $\dot{\theta}$ are shown in Figure 1.4a. Both θ and $\dot{\theta}$ are sinusoidal signals. Now, suppose that we take a guess at the initial conditions of the state variables as $\hat{x}_1(0) = 0.3$ and $\hat{x}_2(0) = 0$. By using the state-space model (1.48), the estimates of θ and $\dot{\theta}$ are calculated and shown in comparison to the true trajectories (see Figure 1.4a). It is seen that the estimate of θ , denoted \hat{x}_1 , is not close to the true θ (see the top plot of Figure 1.4a). This study demonstrated that using the model alone is not sufficient to predict the angle of the pendulum.

Let us design and implement an observer to predict the angle of the pendulum. Assume that the observer gain $K_{ob} = [j_1 \ j_2]^T$. The closed-loop characteristic polynomial for the observer is

$$\begin{aligned} & \det(\lambda I - \begin{bmatrix} 0.9801 & 0.0993 - j_1 \\ -0.3973 & 0.9801 - j_2 \end{bmatrix}) \\ &= (\lambda - 0.9801)(\lambda + j_2 - 0.9801) - 0.3973 \times (j_1 - 0.0993), \end{aligned}$$

which is made to be equal to the desired closed-loop characteristic polynomial $(\lambda - 0.1)(\lambda - 0.2)$. Namely,

$$(\lambda - 0.9801)(\lambda + j_2 - 0.9801) - 0.3973 \times (j_1 - 0.0993) = (\lambda - 0.1)(\lambda - 0.2).$$

Solution of the polynomial equation gives us the observer gain as $j_1 = -1.6284$ and $j_2 = 1.6601$. The estimation of angle is carried out using the observer equation:

$$\begin{bmatrix} \hat{x}_1(k+1) \\ \hat{x}_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9801 & 0.0993 \\ -0.3973 & 0.9801 \end{bmatrix} \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \end{bmatrix} + K_{ob}(x_2(k) - \hat{x}_2(k)), \quad (1.58)$$

with initial condition $\hat{x}_1(0) = 0.3$ and $\hat{x}_2(0) = 0$. Figure 1.4b shows that the estimated angle converges to the true angle in about three steps.

1.6.2 Basic Results About Observability

Definition of Observability

A state variable model of a dynamic system is said to be completely observable if, for any sample time k_0 , there exists a sample time $k_1 > k_0$ such that a knowledge of the output $y(k)$ and input $u(k)$ in the time interval $k_0 \leq k \leq k_1$ is sufficient to determine the initial state $x_m(k_0)$ and as a consequence, $x_m(k)$, for all k between k_0 and k_1 . A necessary and sufficient condition for a linear discrete-time system to be completely observable is, if the observability matrix

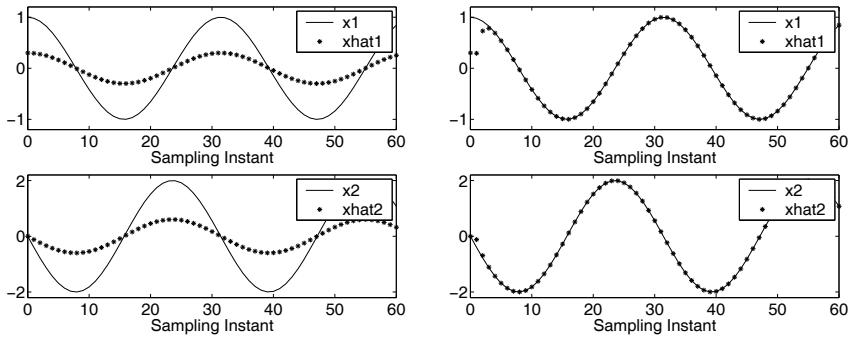


Fig. 1.4. Observer design and implementation

$$L_o = \begin{bmatrix} C_m \\ C_mA_m \\ C_mA_m^2 \\ \vdots \\ C_mA_m^{n-1} \end{bmatrix}$$

has rank n , where n is the dimension of the state variable model.

Example 1.8. A DC motor can be described by a second-order model with an integrator and one time constant (see Figure 1.5). The input is the voltage to the motor and the output is the shaft position. The time constant is due to the mechanical parts of the system. The dynamics due to the electrical parts are neglected because they have small time constants.

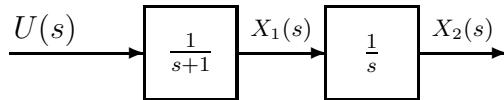


Fig. 1.5. Motor model

By choosing x_1 as the angular velocity and x_2 as the angular position of the motor shaft, we obtain the continuous-time state-space equation:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t). \quad (1.59)$$

Suppose that we take the measurement of rotational speed, leading to

$$y(t) = [1 \ 0] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}.$$

The model is discretized using a sampling interval $\Delta t = 0.1$ to give

$$\begin{aligned} \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} &= \begin{bmatrix} 0.9048 & 0 \\ 0.0952 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.0952 \\ 0.0048 \end{bmatrix} u(k) \\ y(k) &= [1 \ 0] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}. \end{aligned} \quad (1.60)$$

Verify that the discrete-time model (1.60) is not observable, and as a consequence, the closed-loop observer system has a pole at 1.

Solution. The open-loop system has eigenvalues at 0.9048 and 1. The observability matrix is

$$L_o = \begin{bmatrix} C_m \\ C_m A_m \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0.9048 & 0 \end{bmatrix},$$

where $\det(L_o) = 0$. Thus, the pair (C_m, A_m) is not observable. Let us investigate this example further to discover what happens when the system is not observable. Assume that $K_{ob} = [j_1 \ j_2]^T$. Then the closed-loop observer error system is:

$$\tilde{x}(k+1) = \left[\begin{bmatrix} 0.9048 & 0 \\ 0.0952 & 1 \end{bmatrix} - \begin{bmatrix} j_1 \\ j_2 \end{bmatrix} [1 \ 0] \right] \tilde{x}(k). \quad (1.61)$$

Suppose that we put the desired eigenvalues of the observer system at 0.1 and 0.2.³ Then the closed-loop characteristic polynomial is:

$$\det \begin{bmatrix} \lambda - 0.9048 + j_1 & 0 \\ -0.0952 & \lambda - 1 \end{bmatrix} = (\lambda - 0.9048 + j_1)(\lambda - 1). \quad (1.62)$$

In the design of the observer, we let the actual closed-loop characteristic polynomial equal the desired closed-loop characteristic polynomial, and solve for the observer gain vector. In this case, we would let

$$(\lambda - 0.9048 + j_1)(\lambda - 1) = (\lambda - 0.1)(\lambda - 0.2).$$

Note that the second pole at $\lambda = 1$ in (1.62) cannot be moved no matter what choice we make for j_2 , simply because the closed-loop pole is independent of the observer gain. This is the consequence of the pair (C_m, A_m) being unobservable. If the pole that cannot be changed using the observer is asymptotically stable, then the system is not observable, however, it is detectable. In this particular example, the unobservable pole is on the unit circle, thus the system is not even detectable. So, if we measure the angular speed of the motor, then the angular position of the motor cannot be estimated accurately from this measurement.

³ We also call the eigenvalues of the observer system as the closed-loop poles of the observer.

1.6.3 Kalman Filter

If the pair (A_m, C_m) is observable, then for the single-output case, as we discussed, a pole-assignment strategy can be used to determine K_{ob} such that the eigenvalues of the observer (*i.e.*, of the matrix $A_m - K_{ob}C_m$) are at desired locations. For a multi-output system, K_{ob} can be calculated recursively using a Kalman filter. Kalman filters are proposed in a stochastic setting. To this end, we assume that

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) + d(k) \\ y(k) &= C_m x_m(k) + \xi(k), \end{aligned} \quad (1.63)$$

with the covariance matrix of d and ξ , respectively, defined by

$$E\{d(k)d(\tau)^T\} = \Theta \delta(k-\tau)$$

$$E\{\xi(k)\xi(\tau)^T\} = \Gamma \delta(k-\tau),$$

where $\delta(k-\tau) = 1$, if $k = \tau$ and $\delta(k-\tau) = 0$ if $k \neq \tau$.

The optimal observer gain K_{ob} is solved recursively for $i = 0, 1, \dots$, using

$$K_{ob}(i) = A_m P(i) C_m^T (\Gamma + C_m P(i) C_m^T)^{-1}, \quad (1.64)$$

and

$$P(i+1) = A_m \{P(i) - P(i) C_m^T (\Gamma + C_m P(i) C_m^T)^{-1} C_m P(i)\} A_m^T + \Theta. \quad (1.65)$$

More specifically, $P(0)$ satisfies

$$E\{[x(0) - \hat{x}(0)][x(0) - \hat{x}(0)]^T\} = P(0).$$

Assuming that the system (C_m, A_m) is detectable from the output $y(k)$ (*i.e.*, there are no unstable states whose response can not be ‘seen’ from the output) and $(A_m, \Theta^{1/2})$ is stabilizable, then, as $k \rightarrow \infty$, the steady-state solutions of (1.64) and (1.65) satisfy the discrete-time algebraic Riccati equation:

$$P(\infty) = A_m \{P(\infty) - P(\infty) C_m^T (\Gamma + C_m P(\infty) C_m^T)^{-1} C_m P(\infty)\} A_m^T + \Theta, \quad (1.66)$$

and

$$K_{ob}(\infty) = A_m P(\infty) C_m^T (\Gamma + C_m P(\infty) C_m^T)^{-1}. \quad (1.67)$$

Also, the eigenvalues of $A_m - K_{ob}(\infty)C_m$ are guaranteed to be inside the unit circle (*i.e.* stable). To avoid confusion, it is emphasized that the iterative solution of the Riccati equation (1.65) is not required in real time. The observer gain is calculated off-line for predictive control applications.

Kalman filters can be found in the textbooks (see for example, Anderson and Moore, 1979, Grimble and Johnson 1988b, Goodwin, *et al.*, 2000).

1.6.4 Tuning Observer Dynamics

It is often the case that the covariance matrices Θ and Γ , corresponding to the characteristics of the disturbances, are unknown. Thus, in practice, we choose Θ , Γ and an initial $P(0)$ to calculate an observer gain K_{ob} by solving the Riccati equation iteratively until the solution converges to a constant matrix. Then, the closed-loop system obtained is analyzed with respect to the location of eigenvalues contained in $A_m - K_{ob}C_m$, the transient response of the observer, robustness and effect of noise on the response. The elements of the covariance matrices are modified until a desired result is obtained. Such a trial-and-error procedure can be time consuming and frustrating, and is one of the challenges we face when using Kalman-filter-based multivariable system design. In some circumstances, however, it is possible to specify a region in which the closed-loop observer error system poles should reside and to enforce this in the solution. We propose a simple approach, along similar lines to the classic approach in Anderson and Moore (1979), in which the closed-loop observer poles are assigned inside a circle with a pre-specified radius α ($0 < \alpha < 1$). The procedure is summarized as follows. Let the error of the estimated state $\tilde{x}(k) = x(k) - \hat{x}(k)$. Then the observer error system is:

$$\tilde{x}(k+1) = (A_m - K_{ob}C_m)\tilde{x}(k). \quad (1.68)$$

We perform the transformation $\hat{A}_m = \frac{A_m}{\alpha}$ and $\hat{C}_m = \frac{C_m}{\alpha}$ where $0 < \alpha < 1$, leading to a transformed system:

$$\tilde{x}_t(k+1) = \frac{1}{\alpha}(A_m - \hat{K}_{ob}\hat{C}_m)\tilde{x}_t(k) = (\hat{A}_m - \hat{K}_{ob}\hat{C}_m)\tilde{x}_t(k). \quad (1.69)$$

Solving the iterative equations (1.64) and (1.65), or the steady-state Riccati equation (1.66) by using \hat{A}_m and \hat{C}_m to replace A_m and C_m matrices, and then the eigenvalues of $\hat{A}_m - \hat{K}_{ob}(\infty)\hat{C}_m$ are guaranteed to be inside the unit circle (*i.e.*, stable). The resultant observer gain \hat{K}_{ob} is then applied to the original observer system (1.68), leading to the closed-loop characteristic equation:

$$\det(zI - (A_m - \hat{K}_{ob}C_m)) = \det(zI - (\hat{A}_m - \hat{K}_{ob}\hat{C}_m) \times \alpha) = 0. \quad (1.70)$$

Therefore, we conclude that the eigenvalues of $(A_m - \hat{K}_{ob}C_m)$ are equal to the eigenvalues of $\hat{A}_m - \hat{K}_{ob}\hat{C}_m$ multiplied by the factor α , which guarantees that the eigenvalues of the observer error system with \hat{K}_{ob} reside inside the circle of radius α . This procedure makes a direct connection to the observer dynamics via the choice of α . The trial-and-error procedure can be reduced to choose a suitable α along with Θ and Γ to achieve the desired closed-loop performance.

1.7 State Estimate Predictive Control

In the implementation of predictive control, an observer is used for the cases where the state variable $x(k_i)$ at time k_i is not measurable. Essentially, the

state variable $x(k_i)$ is estimated via an observer of the form:

$$\hat{x}(k_i + 1) = A\hat{x}(k_i) + B\Delta u(k_i) + K_{ob}(y(k_i) - C\hat{x}(k_i)). \quad (1.71)$$

Note that in the implementation of predictive control using an observer, the control signal is $\Delta u(k_i)$ and the matrices (A, B, C) come from the augmented model used for the predictive control design. With the information of $\hat{x}(k_i)$ replacing $x(k_i)$, the predictive control law is then modified to find ΔU by minimizing

$$J = (R_s - F\hat{x}(k_i))^T(\bar{R}_s r(k_i) - F\hat{x}(k_i)) - 2\Delta U^T \Phi^T (R_s - F\hat{x}(k_i)) \\ + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U, \quad (1.72)$$

where \bar{R}_s , F , Φ , \bar{R} and ΔU were defined in (1.44 and 1.45).

The optimal solution ΔU is obtained as,

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - F\hat{x}(k_i)). \quad (1.73)$$

Furthermore, application of the receding horizon control principle leads to the optimal solution of $\Delta u(k_i)$ at time k_i :

$$\Delta u(k_i) = K_y r(k_i) - K_{mpc} \hat{x}(k_i), \quad (1.74)$$

which is a standard state feedback control law with estimated $x(k_i)$. The closed-loop state feedback structure is illustrated in Figure 1.6.

What about the closed-loop characteristic equation, and hence the eigenvalues of the closed-loop system? To investigate these issues, we form the closed-loop state-space equation as below:

$$x(k+1) = Ax(k) + B\Delta u(k) \\ = Ax(k) + BK_y r(k) - BK_{mpc} \hat{x}(k), \quad (1.75)$$

where we substituted $\Delta u(k)$ with (1.74). Note that the closed-loop observer error equation is:

$$\tilde{x}(k+1) = (A - K_{ob}C)\tilde{x}(k), \quad (1.76)$$

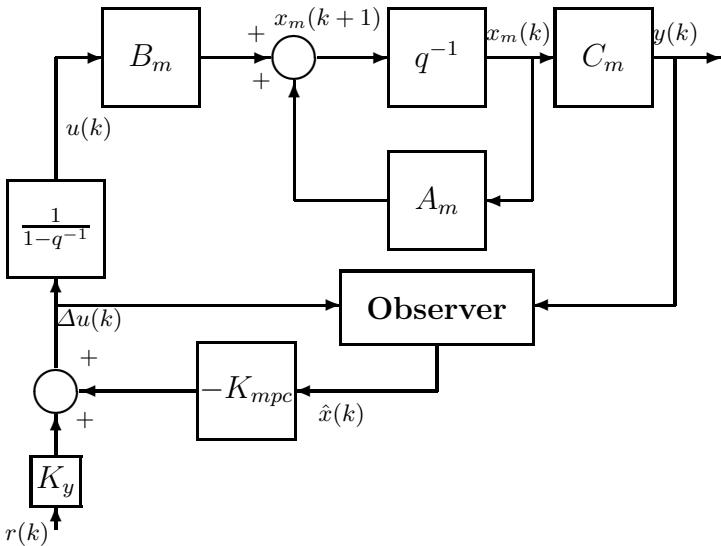
where $\tilde{x}(k) = x(k) - \hat{x}(k)$. Replacing $\hat{x}(k)$ by $x(k) - \tilde{x}(k)$, (1.75) is rewritten as,

$$x(k+1) = (A - BK_{mpc})x(k) - BK_{mpc}\tilde{x}(k) + BK_y r(k). \quad (1.77)$$

Combination of (1.76) with (1.77) leads to:

$$\begin{bmatrix} \tilde{x}(k+1) \\ x(k+1) \end{bmatrix} = \begin{bmatrix} A - K_{ob}C & o_{n \times n} \\ -BK_{mpc} & A - BK_{mpc} \end{bmatrix} \begin{bmatrix} \tilde{x}(k) \\ x(k) \end{bmatrix} + \begin{bmatrix} o_{n \times m} \\ BK_y \end{bmatrix} r(k), \quad (1.78)$$

where $o_{n \times n}$ is a $n \times n$ zero matrix and $o_{n \times m}$ is a $n \times m$ zero matrix. The characteristic equation of the closed-loop state-space system is determined by

**Fig. 1.6.** Block diagram of DMPC with observer

$$\det \left[\lambda I - \begin{bmatrix} A - K_{ob}C & o_{n \times n} \\ -BK_{mpc} & A - BK_{mpc} \end{bmatrix} \right] = 0,$$

which is equivalent to

$$\det(\lambda I - (A - K_{ob}C)) \det(\lambda I - (A - BK_{mpc})) = 0,$$

because the system matrix in (1.78) has a lower block triangular structure. This effectively means that the closed-loop model predictive control system with state estimate has two independent characteristic equations

$$\det(\lambda I - (A - K_{ob}C)) = 0 \quad (1.79)$$

$$\det(\lambda I - (A - BK_{mpc})) = 0. \quad (1.80)$$

Since the closed-loop eigenvalues are the solutions of the characteristic equations, (1.79) and (1.80) indicate that the set of eigenvalues of the combined closed-loop system consists of predictive control-loop eigenvalues and observer-loop eigenvalues. This means that the design of the predictive control law and the observer can be carried out independently (or separately), yet when they are put together in this way, the eigenvalues remain unchanged.

Example 1.9. The augmented model for a double integrated plant (see Example 1.1) is given by

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k), \end{aligned} \quad (1.81)$$

$$\text{where } A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}; B = \begin{bmatrix} 0.5 \\ 1 \\ 0.5 \end{bmatrix}; C = [0 \ 0 \ 1].$$

We will design a state estimate predictive control system and simulate the closed-loop response for a set-point change. The design specifications are $N_c = 5$, $N_p = 30$, the weight on the control signal is $r_w = 10$. The observer is designed using the pole-assignment method where the closed-loop observer poles are 0.01, 0.0105, 0.011, corresponding to a fast dynamic response speed from the observer.

Solution. The open-loop plant has three eigenvalues at 1 where two of these were from the double-integrated plant and one from the predictive controller structure. We use the MATLAB command ‘place’ and write a few lines of MATLAB program to produce the observer gain vector K_{ob} .

```
Pole=[0.01 0.0105 0.011];
K_ob=place(A',C',Pole);
```

where A' , C' are the transposed matrices A^T and C^T . The transposes are needed because the MATLAB program ‘place’ was written for controller design. By using this program, we have used the dual relationship between controller and observer. The resultant observer gain is

$$K_{ob} = [1.9685 \ 0.9688 \ 2.9685]^T.$$

With this set of performance parameters specified, the state feedback control gain is $K_{mpc} = [0.8984 \ 1.3521 \ 0.4039]$, which effectively yields a set of closed-loop eigenvalues at $0.3172 \pm j0.4089$ and 0.3624. Figure 1.7 shows the closed-loop response for a step set-point change. It is seen that the closed-loop output response follows the set-point change, and the control signal converges to zero as the plant has integrators.

1.8 Summary

This chapter has discussed the basic ideas about discrete-time model predictive control. The key terms are: moving horizon window, prediction horizon and control horizon. With the current plant information represented by the state variable vector $x(k_i)$, the prediction of the future behaviour of the plant output relies on the state-space model where the optimal control trajectory is captured by the set of parameters that define the incremental control movement as: $\Delta u(k_i)$, $\Delta u(k_i + 1)$, ..., $\Delta u(k_i + N_c - 1)$. Within the optimization window, the objective of the control system is expressed in terms of the error function between the desired set-point signal and the predicted output signal. With a specific choice of an error function as the measurement of the objective, an optimal solution is obtained for the set of incremental control

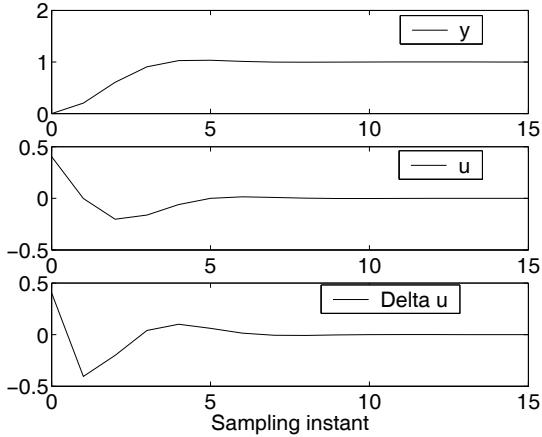


Fig. 1.7. Predictive control of double integrator plant

movement: $\Delta u(k_i)$, $\Delta u(k_i + 1)$, ..., $\Delta u(k_i + N_c - 1)$. Although the optimal control trajectory is calculated for N_c future samples, the implementation of the predictive control uses only the first sample, $\Delta u(k_i)$ while ignoring the rest of the trajectory. The optimization procedure repeats itself when the next sample period arrives. This is based on the receding horizon control principle, where feedback is naturally incorporated in the control system design.

The design model used here is an augmented plant model with embedded integrator(s). By doing so, the control signal to be optimized is the sequence of $\Delta u(k_i + m)$, $m = 0, 1, 2, \dots$, instead of the sequence of the control signal $u(k_i + m)$. An integrator is naturally embedded into the design, leading to the predictive control system tracking constant references and rejecting constant disturbances without steady-state errors. Another significant advantage of this approach is that in implementation, it neither requires the steady-state information about the control ($u(k) = u(k-1) + \Delta u(k)$) nor the information about the steady state of the state variable x_m (Δx_m at steady state is zero). This simplified information requirement becomes more important for a plant having many inputs and many outputs. Because of this formulation of embedding integrators in the design, the model used for prediction has at least one eigenvalue on the unit circle. As a result, it inherits a numerical instability problem when the prediction horizon N_p becomes large. Stability cannot be guaranteed with a small prediction horizon and control horizon parameters, although it can be checked. Parameters N_p and N_c are used as tuning parameters. With some small modifications, as shown in the later chapters of this book (see Chapter 4), this numerical problem is overcome and stability is guaranteed.

There are several major reviews published for MPC that clarify the economic benefits of this class of control algorithms when applied to process industry

(Garcia *et al.*, 1989, Richalet, 1993, Qin and Badgwell, 1996, Morari and Lee, 1999, Mayne *et al.*, 2000). There are also several excellent tutorial papers published in the area of model predictive control (Ricker, 1991, Shah, 1995, Rawlings, 2000, Allgower *et al.*, 1999). Books about predictive control include ‘Adaptive Optimal Control the thinking man’s GPC’ by Bitmead *et al.*, in 1990, ‘Predictive Control’ by Camacho and Bordons in 2004, ‘Predictive Control with Constraints’ by Maciejowski in 2002, and ‘Model-based Predictive Control, a Practical Approach’ by Rossiter in 2003. There is also a book about receding horizon control by Kwon and Han published in 2005.

Problems

1.1. Draw a road map for the contents of this chapter. List the key equations and key concepts at the corresponding locations.

1.2. Imagine yourself as a driver who is trying to steer a vehicle at a bend. Describe your activity in terms of a predictive control system.

1.3. Assume a discrete-time system with input $u(k)$ and output $y(k)$. The system has a constant input disturbance d . Find the augmented state-space model with input $\Delta u(k)$ and output $y(k)$ for the plant model given as below:

$$x_m(k+1) = A_m x_m(k) + B_m u(k) + B_d d; \quad y(k) = C_m x_m(k) \quad (1.82)$$

$$\text{where } A_m = \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & -0.1 \\ 0 & 0 & 0.8 \end{bmatrix}; \quad B_m = \begin{bmatrix} 0.5 \\ 1 \\ -0.6 \end{bmatrix}; \quad C_m = [1 \ 0 \ 1]; \quad B_d = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

1. Calculate the plant transfer function that relates the input $u(k)$ to the output $y(k)$, and the transfer function of the augmented state-space model that relates the input $\Delta u(k)$ to the output $y(k)$.
2. Compare the poles and zeros of these transfer functions.

1.4. Assume that the augmented mathematical model for a discrete-time system is given by

$$x(k+1) = Ax(k) + B\Delta u(k); \quad y(k) = Cx(k), \quad (1.83)$$

$$\text{where } A = \begin{bmatrix} 0.6 & 0 \\ 0.6 & 1 \end{bmatrix}; \quad B = \begin{bmatrix} 0.3 \\ 0.3 \end{bmatrix}; \quad C = [0 \ 1].$$

1. At time $k_i = 0$, assuming control horizon $N_c = 4$, $N_p = 10$ and the initial state variable $x(0) = [0.1 \ 0.2]^T$, express the predicted output

$$y(k_i + 1 | k_i), \quad y(k_i + 2 | k_i), \quad \dots, \quad y(k_i + N_p | k_i)$$

in terms of ΔU where

$$\Delta U = [\Delta u(k_i) \ \Delta u(k_i + 1) \ \Delta u(k_i + 2) \ \Delta u(k_i + 3)]^T.$$

2. With the set-point signal $r(k_i) = 0$, find the optimal control ΔU that minimizes the cost function:

$$J = Y^T Y + \Delta U^T \bar{R} \Delta U,$$

where \bar{R} is a diagonal matrix ($\bar{R} = r_w I$, $r_w = 3$ and I is 4×4 identity matrix). What is the minimum of the cost function, J_{min} ?

3. Reduce r_w to 0, and observe the changes to the optimal control ΔU .
 4. Compare the value of J_{min} for the case of $r_w = 0$ with the value from the previous case when $r_w = 3$.

1.5. A first-order model is often used to describe the dynamics of a liquid vessel, where the input to the system is the flow rate and the output is the fluid level (see Figure 1.8). Assume that a discrete-time first-order transfer

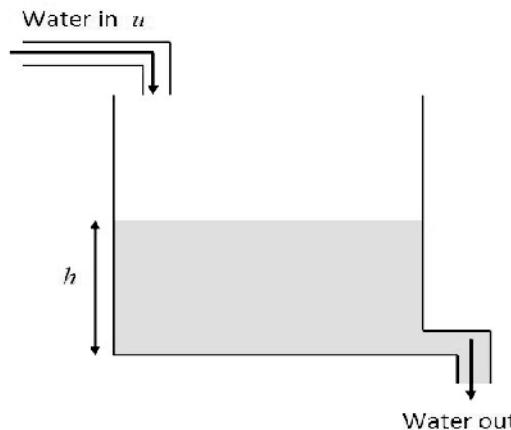


Fig. 1.8. Schematic diagram for a single tank

function is obtained for a fluid system, leading to the relationship between the input and output:

$$Y(z) = \frac{0.01}{z - 0.6} U(z)$$

- Choosing $x_m(k) = h(k) = y(k)$, convert the transfer function model to a state-space model, and design a model predictive control system that will maintain the liquid level at a desired reference position $r(k) = 0.5$. The design parameters for the predictive control system are specified as $N_c = 4$, $N_p = 16$, $\bar{R} = I$.
- With initial condition $x(0) = [0 \ 0]^T$, simulate the closed-loop predictive control system using receding horizon control. (Hint: the state variables are $x_1(k) = y(k) - y(k - 1)$ and $x_2(k) = y(k)$. Both are measurable.)

1.6. Time delay in a discrete-time system appears as part of the denominator of the transfer function. Hence the number of delays will increase the dimensionality of the state-space model. Assume that a discrete-time system is described by the z transfer function

$$G(z) = \frac{0.1z^{-5}}{1 - 1.4z^{-1} + 0.48z^{-2}}.$$

This is a time-delay system. An alternative representation of this transfer function is

$$G(z) = \frac{0.1}{z^3(z^2 - 1.4z + 0.48)},$$

indicating that the system is of fifth order (the number of poles determines the model order). A state-space model for this transfer function is

$$x_m(k+1) = A_m x_m(k) + B_m u(k); \quad y(k) = C_m x_m(k) \quad (1.84)$$

$$\text{where } A_m = \begin{bmatrix} 1.4 & -0.48 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}; \quad B_m = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}; \quad C_m = [0 \ 0 \ 0 \ 0 \ 0.1].$$

1. Design a predictive control system that will track a unit step reference signal. It takes about 20 samples for the step response of this system to reach steady state. Assuming that the closed-loop response will be faster than the open-loop response, choose the prediction horizon $N_p = 16$ and the control horizon $N_c = 4$. The choice of weight matrix \bar{R} could interact with the choices of N_p and N_c . Here, select $\bar{R} = r_w I$ with $r_w = 0.01$. For a larger diagonal element r_w in \bar{R} , the prediction horizon may be larger.
2. With the selection of design parameters, following the Tutorial 1.2, calculate the feedback gain of the predictive control system and the closed-loop eigenvalues.
3. With zero initial condition at $x(0)$ and a unit step input signal at time $k = 0$, simulate the closed-loop response by following the Tutorial 1.3.

1.7. A discrete-time signal $f_0(k) = \beta(k - 10)^2$ is corrupted by white noise $\epsilon(k)$, where β is unknown. Design an observer that will estimate the unknown coefficient β from the noisy measurement $f(k) = f_0(k) + \epsilon(k)$. The closed-loop poles of the observer are placed at 0.1, 0.15, 0.2. (Hint: choose $x_1(k) = \beta(k - 10)^2$; $x_2(k) = \beta(k - 10)$; $x_3(k) = \beta$.)

1.8. A severely under-damped mechanical system is described by the continuous-time transfer function model $G(s)$, where

$$G(s) = \frac{0.1}{s^2 + 2\xi\omega_0 s + \omega_0^2}$$

with the damping coefficient $\xi = 0.001$ and $\omega_0 = 1$.

1. Discretize the continuous-time transfer function using sampling interval $\Delta t = 0.5$ to obtain the discrete-time transfer function.
2. Design a predictive control system with observer for rejecting constant input disturbance and following a step set-point change with zero steady-state error. Because the open-loop system is severely under-damped, large prediction horizon and control horizon are required to achieve desired performance. Choose the prediction horizon $N_p = 60$ and the control horizon $N_c = 20$. The weight matrix $\bar{R} = r_w I$ with $r_w = 0.1$. The observer poles are placed at 0.1, 0.2, 0.3.
3. Simulate the closed-loop performance with a unit step input set-point signal at $k = 0$ and a unit step input disturbance entering the system at $k = 100$.

1.9. Robustness of a predictive control system against plant unmodelled dynamics is a very important aspect of the design. Without changing the predictive control system, introduce a time-delay of 3 samples to the plant transfer function $G(z)$ in Problem 1.8. Simulate the closed-loop performance with the identical conditions. Is the closed-loop predictive control system stable? If not, go back to the original Problem 1.8 and increase r_w in the weight matrix until the closed-loop system is stable. Analyze your observations.

1.10. Verify that the state-space model of a DC motor is observable when the position of the motor shaft is taken as the measurement and the state-space equation is

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9048 & 0 \\ 0.0952 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.0952 \\ 0.0048 \end{bmatrix} u(k)$$

$$y(k) = [0 \ 1] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}, \quad (1.85)$$

where x_1 is the velocity and x_2 is the position of the motor shaft.

1. Augment the discrete-time plant model with an integrator, and check controllability and observability of the augmented model.
2. Choosing $N_p = 10$, $N_c = 4$, and $\bar{R} = 0.1I$, calculate the gain matrices K_y and K_{mpc} for the predictive control system. Where are the closed-loop eigenvalues?
3. Design an observer, positioning the closed-loop observer poles at 0.1, 0.2 and 0.3.
4. Simulate the state estimate predictive control system, using the set-point signal $r(k) = 2$ for all k . Now, create an input voltage drop by adding a constant $d = -0.5$ to the control signal at $k = 20$, and simulate how the predictive control system rejects the disturbance and maintains the output at set-point $r = 2$.

Discrete-time MPC with Constraints

2.1 Introduction

This chapter discusses discrete-time model predictive control with constraints. The chapter begins with a motivational example to illustrate how the performance of a control system can deteriorate significantly when the control signals from the original design meet with operational constraints. The example also shows that with a small modification, the degree of performance deterioration can be reduced if the constraints are incorporated in the implementation, leading to the idea of constrained control. Then, the chapter reveals how to formulate the constrained control problem in the context of predictive control, which essentially becomes a quadratic programming problem. Assuming that most readers have not studied quadratic programming before, a section is devoted to introducing the fundamentals of quadratic programming and presenting the solutions with simple and effective numerical algorithms. The final section of this chapter shows several examples of constrained control problems.

2.2 Motivational Examples

Before we begin our study on constrained control, let us look at an example where the control system operates with and without control signal saturation limits.

Example 2.1. A mathematical model for an undamped oscillator is given by

$$\begin{aligned} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -4 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \\ y(t) &= [0 \ 1] \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}. \end{aligned} \tag{2.1}$$

With sampling interval $\Delta t = 0.1$, the corresponding discrete-time state-space model is

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} 0.9801 & 0.0993 \\ -0.3973 & 0.9801 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.0993 \\ -0.0199 \end{bmatrix} u(k) \quad (2.2)$$

$$y(k) = [0 \ 1] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}. \quad (2.3)$$

Suppose that the design objective is to design the predictive control system such that the output of the plant has to track a unit step reference signal as fast as possible. To this end, we select the prediction horizon $N_p = 10$ and the control horizon $N_c = 3$. There is no weight on the control signal, *i.e.*, $\bar{R} = 0$. Examine what happens if the control amplitude is limited to ± 25 by saturation.

Solution. The data matrices in the predictive control system are

$$\Phi^T \Phi = \begin{bmatrix} 6.0067 & 4.8853 & 3.8150 \\ 4.8853 & 4.0013 & 3.1475 \\ 3.8150 & 3.1475 & 2.4952 \end{bmatrix}; \Phi^T F = \begin{bmatrix} 65.5285 & -25.2099 & -6.1768 \\ 53.1281 & -19.6709 & -4.7606 \\ 41.3553 & -14.6974 & -3.5334 \end{bmatrix}$$

$$\Phi^T \bar{R}s = \begin{bmatrix} -6.1768 \\ -4.7606 \\ -3.5334 \end{bmatrix}.$$

The state feedback gain matrix is

$$K_{mpc} = [17.9064 \ -39.0664 \ -29.9659].$$

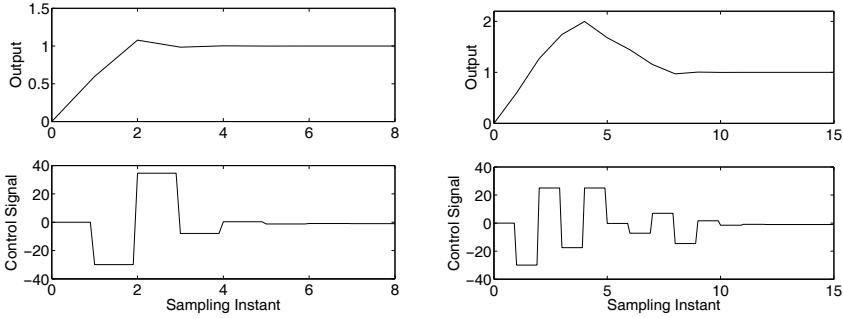
The closed-loop eigenvalues are at $-0.1946, 0, 0$. An observer is designed with poles positioned at $0, 0, 0$. In the following, we look at two cases: without control saturation; and with control signal saturation.

Case A. Without control saturation

The closed-loop response is illustrated in Figure 2.1a. It is seen that the output converges to the set-point signal after 4 samples. Indeed, the design objective has been achieved. If the control amplitude is of concern, then we note that this optimal control has a large amplitude that is close to 40 at its maximum.

Case B. With control saturation

Assume that the control amplitude has limits at ± 25 due to operational constraint, namely, $-25 \leq u(k) \leq 25$. Then, this limit prevents the control signal from being implemented to the plant when its amplitude exceeds this limit. Thus, $u(k) = 25$, if $u(k) > 25$; and $u(k) = -25$ if $u(k) < -25$. When this



(a) Closed-loop response without constraint (b) Closed-loop control with saturation

Fig. 2.1. Comparison of responses with and without constraints

happens, the closed-loop performance significantly deteriorates, as shown in Figure 2.1b. The figure shows that when the control signal comes out of the saturation after two sample periods, it becomes oscillatory, and as a result, the plant output has a significant over-shoot.

This example illustrated that if we do not pay attention to the saturation of the control, then in the presence of constraints, the closed-loop control performance could severely deteriorate. From this example, it is obvious that it is important to find a way to deal with the problem when the control signal becomes saturated. The next example shows that a small modification in the predictive control law will enable the system to handle the constraint without significant performance deterioration.

Example 2.2. A common practice in dealing with saturation is to let the model know the difference in $\Delta u(k)$ when saturation becomes effective. Continue with Example 2.1 with a modification of the control calculation where the difference of the control signal $\Delta u(k)$ is taken into consideration in the presence of the constraint.

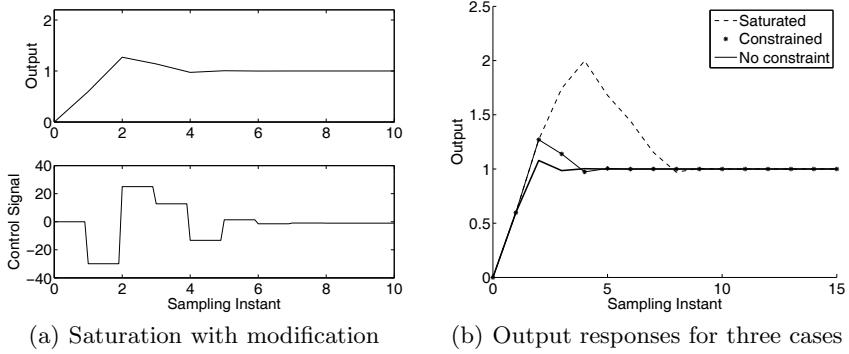
Solution. The small modification is to calculate $\Delta u(k_i)$ in the following way. If the calculated control $u(k_i) > 25$, then

$$\begin{aligned} u(k_i) &= 25 \text{ and} \\ \Delta u(k_i) &= 25 - u(k_i - 1). \end{aligned} \quad (2.4)$$

If the calculated control $u(k_i) < -25$, then

$$\begin{aligned} u(k_i) &= -25 \text{ and} \\ \Delta u(k_i) &= -25 - u(k_i - 1). \end{aligned} \quad (2.5)$$

This new $\Delta u(k_i)$ is used in the observer to predict the next sample of state variable $\hat{x}(k_i + 1)$:

**Fig. 2.2.** Comparison of results when constraints are present

$$\hat{x}(k_i + 1) = A\hat{x}(k_i) + B\Delta u(k_i) + K_{ob}(y(k_i) - C\hat{x}(k_i)). \quad (2.6)$$

The calculation is performed for all sample points. This small modification results in a significant change in the control system performance. Figure 2.2a shows the improvement of the control system in the presence of activated constraint. What we notice is that the control signal comes out of saturation without oscillation. As a result, the over-shoot in the closed-loop response (as shown in Figure 2.2b of Example 2.1) is significantly reduced. For comparison purpose, the output responses are presented in Figure 2.2b for three different cases, where Case A is the output response without saturation (solid line); Case B is the output response with saturation (dotted); and Case C is the output response with modified control saturation (dotted+solid). The improvement on closed-loop performance with this modified saturation is further illustrated by the comparative data in Table 2.1. As we will learn later in the chapter, this solution is in fact the optimal solution in the context of predictive control with amplitude constraint for a single-input, single-output system.

There are several issues to note from these examples. The saturation on the control signal can make the control system performance deteriorate signifi-

Table 2.1. Comparison of performance parameters. Predictive control system with saturation and predictive control system with modified saturation

	With control saturation	With modified control saturation
y_{max}	2.01	1.25
Over-shoot(%)	100	25
No. Samples to settling	11	6
u^{max}	25	25
u^{min}	-25	-25

cantly. The modification to overcome control saturation effects is to calculate the value $\Delta u(k)$ when the saturation is reached, and use this information to modify the predicted state variables. Because it was a single-input, single-output system and only two constraints were imposed, this small modification was feasible. However, for a multi-input, multi-output system, the limits of the system operation appear in many forms, such as the limits on each control signal and its difference $\Delta u(k)$, as well as on the state variables and output variables. It is a much more complex task to work out the individual saturation limits in a co-ordinated manner. Even if we could do this, the solution still would not guarantee the optimal performance of the multi-input, multi-output system. It is on these grounds that we propose a constrained control framework using the model predictive control system. The strength of the approach lies in the optimality that it achieves in a systematic manner, and the flexibility/generality to cope with a multi-input, multi-out system with various constraints. Perhaps above all, this simplicity in concept is readily accepted by application engineers.

2.3 Formulation of Constrained Control Problems

The core idea in Example 2.2 was to modify $\Delta u(k)$ to suit the situation when the constraint became activated. In the context of predictive control, this problem is handled systematically by using optimization. To this end, we need to formulate the predictive control problem as an optimization problem that takes into account the constraints present.

This section discusses the operational constraints that are frequently encountered in the design of control systems. These operational constraints are presented as linear inequalities of the control and plant variables.

2.3.1 Frequently Used Operational Constraints

There are three major types of constraints frequently encountered in applications. The first two types deal with constraints imposed on the control variables $u(k)$, and the third type of constraint deals with output $y(k)$ or state variable $x(k)$ constraints. For clarity, we will discuss single-input, single-output systems first and subsequently extend the cases to multi-input, multi-output systems.

Constraints on the Control Variable Incremental Variation

These are hard constraints on the size of the control signal movements, *i.e.*, on the rate of change of the control variables ($\Delta u(k)$). Suppose that for a single-input system, the upper limit is Δu^{max} and the lower limit is Δu^{min} . The constraints are specified in the form

$$\Delta u^{min} \leq \Delta u(k) \leq \Delta u^{max}. \quad (2.7)$$

Note that we use *less than plus equal to* in (2.7), where the equality will play the critical role in the solution of the constrained control problem (see later sections).

The rate of change constraints can be used to impose directional movement constraints on the control variables; for instance, if $u(k)$ can only increase, not decrease, then possibly, we select $0 \leq \Delta u(k) \leq \Delta u^{max}$. The constraint on $\Delta u(k)$ can be used to cope with the cases where the rate of change of the control amplitude is restricted or limited in value. For example, in a control system implementation, assuming that the control variable $u(k)$ is only permitted to increase or decrease in a magnitude less 0.1 unit, then the operational constraint is

$$-0.1 \leq \Delta u(k) \leq 0.1.$$

Constraints on the Amplitude of the Control Variable

These are the most commonly encountered constraints among all constraint types. For instance, we cannot expect a valve to open more than 100 percent nor a voltage to go beyond a given range. These are the physical hard constraints on the system. Simply, we demand that

$$u^{min} \leq u(k) \leq u^{max}.$$

Here, we need to pay particular attention to the fact that $u(k)$ is an incremental variable, not the actual physical variable. The actual physical control variable equals the incremental variable u plus its steady-state value u_{ss} . A common mistake is to mix these two. For instance, if a valve is allowed to open in the range between 15% and 80% and the valve's normal operating value is at 30%, then $u^{min} = 15\% - 30\% = -15\%$ and $u^{max} = 80\% - 30\% = 50\%$.

Output Constraints

We can also specify the operating range for the plant output. For instance, supposing that the output $y(k)$ has an upper limit y^{max} and a lower limit y^{min} , then the output constraints are specified as

$$y^{min} \leq y(k) \leq y^{max}. \quad (2.8)$$

Output constraints are often implemented as ‘soft’ constraints in the way that a slack variable $s_v > 0$ is added to the constraints, forming

$$y^{min} - s_v \leq y(k) \leq y^{max} + s_v. \quad (2.9)$$

There is a primary reason why we use a slack variable to form ‘soft’ constraints for output. Output constraints often cause large changes in both the

control and incremental control variables when they are enforced (we term them *become active* in the later sections). When that happens, the control or incremental control variables can violate their own constraints and the problem of constraint conflict occurs. In the situations where the constraints on the control variables are more essential to plant operation, the output constraints are often relaxed by selecting a larger slack variable s_v to resolve the conflict problem.

Similarly, we can impose constraints on the state variables if they are measurable or impose the constraints on observer state variables. They also need to be in the form of ‘soft’ constraints for the same reasons as the output case above.

Constraints in a Multi-input and Multi-output Setting

If there is more than one input, then the constraints are specified for each input independently. In the multi-input case, suppose that the constraints are given for the upper limits as

$$[\Delta u_1^{max} \Delta u_2^{max} \dots \Delta u_m^{max}],$$

and lower limits as

$$[\Delta u_1^{min} \Delta u_2^{min} \dots \Delta u_m^{min}].$$

Each variable with rate of change is specified as

$$\begin{aligned} \Delta u_1^{min} &\leq \Delta u_1(k) \leq \Delta u_1^{max} \\ \Delta u_2^{min} &\leq \Delta u_2(k) \leq \Delta u_2^{max} \\ &\vdots \\ \Delta u_m^{min} &\leq \Delta u_m(k) \leq \Delta u_m^{max}. \end{aligned} \tag{2.10}$$

Similarly, suppose that the constraints are given for the upper limit of the control signal as

$$[u_1^{max} u_2^{max} \dots u_m^{max}],$$

and lower limit as

$$[u_1^{min} u_2^{min} \dots u_m^{min}].$$

Then, the amplitude of each control signal is required to satisfy the constraints:

$$\begin{aligned} u_1^{min} &\leq u_1(k) \leq u_1^{max} \\ u_2^{min} &\leq u_2(k) \leq u_2^{max} \\ &\vdots \\ u_m^{min} &\leq u_m(k) \leq u_m^{max}. \end{aligned} \tag{2.11}$$

Similarly, constraints are specified for each output and state variable if they are required. In short, the constraints for a multi-input and multi-output system are specified for each input and output independently.

2.3.2 Constraints as Part of the Optimal Solution

Having formulated the constraints as part of the design requirements, the next step is to translate them into linear inequalities, and relate them to the original model predictive control problem. The key here is to parameterize the constrained variables using the same parameter vector ΔU as the ones used in the design of predictive control. Therefore, the constraints are expressed in a set of linear equations based on the parameter vector ΔU . The vector ΔU is often called the *decision variable* in optimization literature. Since the predictive control problem is formulated and solved in the framework of receding horizon control, the constraints are taken into consideration for each moving horizon window. This allows us to vary the constraints at the beginning of each optimization window, and also gives us the means to tackle the constrained control problem numerically. Based on this idea, if we want to impose the constraints on the rate of change of the control signal $\Delta u(k)$ at time k_i , the constraints at sample time k_i are expressed as

$$\Delta u^{min} \leq \Delta u(k_i) \leq \Delta u^{max}.$$

From the time instance k_i , the predictive control scheme looks into the future. The constraints at future samples, for example on the first three samples, $\Delta u(k_i), \Delta u(k_i + 1), \Delta u(k_i + 2)$ are imposed as

$$\Delta u^{min} \leq \Delta u(k_i) \leq \Delta u^{max}$$

$$\Delta u^{min} \leq \Delta u(k_i + 1) \leq \Delta u^{max}$$

$$\Delta u^{min} \leq \Delta u(k_i + 2) \leq \Delta u^{max}.$$

In principle, all the constraints are defined within the prediction horizon. However, in order to reduce the computational load, we sometimes choose a smaller set of sampling instants at which to impose the constraints, instead of all the future samples. The following example shows how to express the constraints from the design specification in terms of a function of ΔU .

Example 2.3. In the motor control system, suppose that the input voltage variation is limited to 2 V and 6 V. The steady state of the control signal is at 4 V. Assuming that the control horizon is selected to be $N_c = 4$, express the constraint on $\Delta u(k_i)$ and $\Delta u(k_i + 1)$ in terms of ΔU for the first two sample times.

Solution. The parameter vector to be optimized in the predictive control system at time k_i is $\Delta U = [\Delta u(k_i) \Delta u(k_i + 1) \Delta u(k_i + 2) \Delta u(k_i + 3)]^T$.

Note that

$$u(k_i) = u(k_i - 1) + \Delta u(k_i) = u(k_i - 1) + [1 \ 0 \ 0 \ 0] \Delta U \quad (2.12)$$

$$\begin{aligned} u(k_i + 1) &= u(k_i) + \Delta u(k_i + 1) = u(k_i - 1) + \Delta u(k_i) + \Delta u(k_i + 1) \\ &= u(k_i - 1) + [1 \ 1 \ 0 \ 0] \Delta U. \end{aligned} \quad (2.13)$$

With the limits on the control variables, by subtracting the steady-state value of the control, as $u^{min} = 2 - 4 = -2$ and $u^{max} = 6 - 4 = 2$, the constraints are expressed as

$$\begin{bmatrix} -2 \\ -2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(k_i - 1) + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \Delta U \leq \begin{bmatrix} 2 \\ 2 \end{bmatrix}. \quad (2.14)$$

Now, since we have expressed the constraints as the inequalities with linear-in-the-parameter ΔU , the next step is to combine the constraints with the original cost function J used in the design of predictive control. As the optimal solutions will be obtained using quadratic programming, the constraints need to be decomposed into two parts to reflect the lower limit, and the upper limit with opposite sign. Namely, for instance, the constraints

$$\Delta U^{min} \leq \Delta U \leq \Delta U^{max}$$

will be expressed by two inequalities:

$$-\Delta U \leq -\Delta U^{min} \quad (2.15)$$

$$\Delta U \leq \Delta U^{max}. \quad (2.16)$$

In a matrix form, this becomes

$$\begin{bmatrix} -I \\ I \end{bmatrix} \Delta U \leq \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix}. \quad (2.17)$$

This procedure applies to all the constraints mentioned in this section, including control and output constraints.

Traditionally, the constraints are imposed for all future sampling instants, and all constraints are expressed in terms of the parameter vector ΔU . In the case of a manipulated variable constraint, we write:

$$\begin{bmatrix} u(k_i) \\ u(k_i + 1) \\ u(k_i + 2) \\ \vdots \\ u(k_i + N_c - 1) \end{bmatrix} = \begin{bmatrix} I \\ I \\ I \\ \vdots \\ I \end{bmatrix} u(k_i - 1) + \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ I & I & 0 & \dots & 0 \\ I & I & I & \dots & 0 \\ \vdots & & & & \\ I & I & \dots & I & I \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i + 1) \\ \Delta u(k_i + 2) \\ \vdots \\ \Delta u(k_i + N_c - 1) \end{bmatrix}. \quad (2.18)$$

Re-writing (2.18) in a compact matrix form, with C_1 and C_2 corresponding to the appropriate matrices, then the constraints for the control movement are imposed as,

$$-(C_1 u(k_i - 1) + C_2 \Delta U) \leq -U^{min} \quad (2.19)$$

$$(C_1 u(k_i - 1) + C_2 \Delta U) \leq U^{max}, \quad (2.20)$$

where U^{min} and U^{max} are column vectors with N_c elements of u^{min} and u^{max} , respectively. Similarly, for the increment of the control signal, we have the constraints:

$$-\Delta U \leq -\Delta U^{min} \quad (2.21)$$

$$\Delta U \leq \Delta U^{max}, \quad (2.22)$$

where ΔU^{min} and ΔU^{max} are column vectors with N_c elements of Δu^{min} and Δu^{max} , respectively. The output constraints are expressed in terms of ΔU :

$$Y^{min} \leq Fx(k_i) + \Phi \Delta U \leq Y^{max}. \quad (2.23)$$

Finally, the model predictive control in the presence of hard constraints is proposed as finding the parameter vector ΔU that minimizes

$$J = (R_s - Fx(k_i))^T (R_s - Fx(k_i)) - 2\Delta U^T \Phi^T (R_s - Fx(k_i)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U, \quad (2.24)$$

subject to the inequality constraints

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} \Delta U \leq \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix}, \quad (2.25)$$

where the data matrices are

$$M_1 = \begin{bmatrix} -C_2 \\ C_2 \end{bmatrix}; \quad N_1 = \begin{bmatrix} -U^{min} + C_1 u(k_i - 1) \\ U^{max} - C_1 u(k_i - 1) \end{bmatrix}; \quad M_2 = \begin{bmatrix} -I \\ I \end{bmatrix};$$

$$N_2 = \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix}; \quad M_3 = \begin{bmatrix} -\Phi \\ \Phi \end{bmatrix}; \quad N_3 = \begin{bmatrix} -Y^{min} + Fx(k_i) \\ Y^{max} - Fx(k_i) \end{bmatrix}.$$

The matrix $\Phi^T \Phi + \bar{R}$ is the Hessian matrix and is assumed to be positive definite. Since the cost function J is a quadratic, and the constraints are linear inequalities, the problem of finding an optimal predictive control becomes one of finding an optimal solution to a standard quadratic programming problem. For compactness of expression, we denote (2.25) by

$$M \Delta U \leq \gamma, \quad (2.26)$$

where M is a matrix reflecting the constraints, with its number of rows equal to the number of constraints and number of columns equal to the dimension of ΔU . When the constraints are fully imposed, the number of constraints is equal to $4 \times m \times N_c + 2 \times q \times N_p$, where m is the number of inputs and q is the number of outputs. The total number of constraints is, in general, greater than the dimension of the decision variable ΔU . Because the receding horizon control law implements the first control movement and ignores the rest of the calculated future control signals, a question naturally arises as to whether it is necessary to impose constraints on all future trajectories of both the control signals and system output. This question will be investigated in a later section of this chapter.

2.4 Numerical Solutions Using Quadratic Programming

The standard quadratic programming problem has been extensively studied in the literature (see for example, Luenberger, 1984, Fletcher, 1981, Boyd and Vandenberghe, 2004). Since this is a field of study in its own right, it requires a considerable effort to completely understand the relevant theory and algorithms. The required numerical solution for MPC is often viewed as an obstacle in the application of MPC. However, what we can do here is to understand the essence of quadratic programming so that we can produce the essential computational programs required. The advantage of doing so is that we can access the code if anything goes wrong; we can also write safety 'jacket' software for real-time applications. These aspects are very important in an industrial environment. To be consistent with the literatures of quadratic programming, the decision variable is denoted by x . The objective function J and the constraints are expressed as

$$J = \frac{1}{2}x^TEx + x^TF \quad (2.27)$$

$$Mx \leq \gamma, \quad (2.28)$$

where E , F , M and γ are compatible matrices and vectors in the quadratic programming problem. Without loss of generality, E is assumed to be symmetric and positive definite.

2.4.1 Quadratic Programming for Equality Constraints

The simplest problem of quadratic programming is to find the constrained minimum of a positive definite quadratic function with linear equality constraints. Each linear equality constraint defines a hyperplane. Positive definite quadratic functions have their level surfaces as hyperellipsoids. Intuitively, the constrained minimum is located at the point of tangency between the boundary of the feasible set and the minimizing hyperellipsoid. Further illustration is given by the following example.

Example 2.4. Minimize

$$J = (x_1 - 2)^2 + (x_2 - 2)^2,$$

subject to

$$x_1 + x_2 = 1.$$

Solution. The global minimum, without constraint, is at

$$x_1 = 2; x_2 = 2.$$

The feasible solutions are the combinations of x_1 and x_2 that satisfy the linear equality. From the constraint, the feasible solution x_2 is expressed as

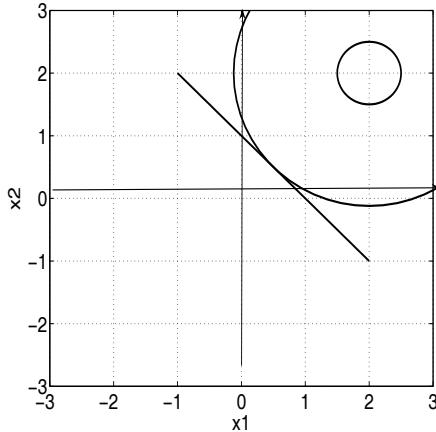


Fig. 2.3. Illustration of constrained optimal solution

$$x_2 = 1 - x_1.$$

Substituting this into the objective function, we obtain

$$\begin{aligned} J &= (x_1 - 2)^2 + (1 - x_1 - 2)^2 \\ &= 2x_1^2 - 2x_1 + 5. \end{aligned} \quad (2.29)$$

In order to minimize J , the derivative $\frac{\partial J}{\partial x_1} = 4x_1 - 2 = 0$, giving the minimizing solution $x_1 = 0.5$. Now, from the constraint equation, $x_2 = 1 - x_1 = 0.5$.

Figure 2.3 illustrates the optimal solution on (x_1, x_2) plane, where the equality constraint defines the straight line and the positive definite quadratic function has its level surface as circles and the constrained minimum is located at the point of tangency between the straight line and the minimizing circle, which is $x_1 = 0.5$ and $x_2 = 0.5$.

The solution of Example 2.4 is easy to understand and it demonstrated the location of the constrained minimum. We consider a general approach to the constrained optimization with equality constraints.

Lagrange Multipliers

To minimize the objective function subject to equality constraints, let us consider the so-called Lagrange expression

$$J = \frac{1}{2}x^T Ex + x^T F + \lambda^T(Mx - \gamma). \quad (2.30)$$

It is easy to see that the value of (2.30) subject to the equality constraints $Mx = \gamma$ being satisfied is the same as the original objective function. We

now consider (2.30) as an objective function in $n + m$ variables x and λ , where n is the dimension of x and m is the dimension of λ . The procedure of minimization is to take the first partial derivatives with respect to the vectors x and λ , and then equate these derivatives to zero. This gives us the results

$$\frac{\partial J}{\partial x} = Ex + F + M^T \lambda = 0 \quad (2.31)$$

$$\frac{\partial J}{\partial \lambda} = Mx - \gamma = 0. \quad (2.32)$$

The linear equations (2.31) together with (2.32) contain $n + m$ variables x and λ , which are the necessary conditions for minimizing the objective function with equality constraints. The elements of the vector λ are called Lagrange multipliers.

The minimization of the Lagrange expression is straightforward. The optimal λ and x are found via the set of linear equations defined by (2.31) and (2.32) where

$$\lambda = -(ME^{-1}M^T)^{-1}(\gamma + ME^{-1}F) \quad (2.33)$$

$$x = -E^{-1}(M^T \lambda + F). \quad (2.34)$$

It is interesting to note that (2.34) can be written as two terms:

$$x = -E^{-1}F - E^{-1}M^T \lambda = x^0 - E^{-1}M^T \lambda,$$

where the first term $x^0 = -E^{-1}F$ is the global optimal solution that will give a minimum of the original cost function J without constraints, and the second term is a correction term due to the equality constraints.

The following example is used to illustrate the minimization with equality constraints.

Example 2.5. Minimize

$$J = \frac{1}{2}x^T Ex + x^T F,$$

subject to

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ 3x_1 - 2x_2 - 3x_3 &= 1, \end{aligned} \quad (2.35)$$

$$\text{where } E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; F = \begin{bmatrix} -2 \\ -3 \\ -1 \end{bmatrix}.$$

Solution. Without the equality constraints, the optimal solution is

$$x^0 = -E^{-1}F = [2 \ 3 \ 1]^T.$$

Writing the two equality constraints given by (2.35) in matrix form, we obtain the M and γ matrices as

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 3 & -2 & -3 \end{bmatrix}; \quad \gamma = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

To use (2.33) the following quantities are required

$$ME^{-1}M^T = \begin{bmatrix} 3 & -2 \\ -2 & 22 \end{bmatrix}; \quad ME^{-1}F = \begin{bmatrix} -6 \\ 3 \end{bmatrix}.$$

Note that the determinant $\det(ME^{-1}M^T) = 62$, thus the matrix $ME^{-1}M^T$ is invertible. The λ vector is

$$\lambda = -(ME^{-1}M^T)^{-1}(\gamma + ME^{-1}F) = \begin{bmatrix} 1.6452 \\ -0.0323 \end{bmatrix}.$$

The x vector that minimizes the objective function is

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x^0 - E^{-1}M^T\lambda = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 3 \\ 1 & -2 \\ 1 & -3 \end{bmatrix} \begin{bmatrix} 1.6452 \\ -0.0323 \end{bmatrix} = \begin{bmatrix} 0.4516 \\ 1.2903 \\ -0.7419 \end{bmatrix}.$$

Example 2.6. In this example, we examine what happens to the constrained optimal solution when the linear constraints are dependent. We assume that the objective function

$$J = \frac{1}{2}x^TEx + x^TF, \quad (2.36)$$

where the matrices $E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $F = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$ and the constraints are

$$\begin{aligned} x_1 + x_2 &= 1 \\ 2x_1 + 2x_2 &= 6. \end{aligned} \quad (2.37)$$

Use graphs to demonstrate that there is no feasible solution of x_1 and x_2 that will satisfy the equality constraints (2.37). In addition, demonstrate that the matrix $M^TE^{-1}M$ is not invertible.

Solution. Figure 2.4 shows that the two equality constraints are defined by two parallel lines on the (x_1, x_2) plane. Because the lines do not intersect, there does not exist a feasible set of parameters x_1 and x_2 that will simultaneously satisfy both linear equations. We can also examine what happens to the Lagrange multipliers when there is no feasible solution. The M and γ matrices are

$$M = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}; \quad \gamma = \begin{bmatrix} 1 \\ 6 \end{bmatrix}.$$

Note that because of the linear dependency in the constraints, the matrix $ME^{-1}M^T = \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix}$ has a zero determinant, and does not have an inverse.

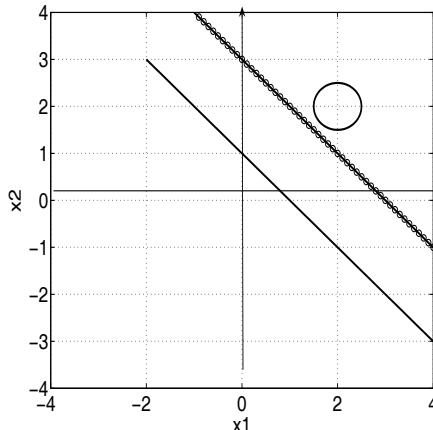


Fig. 2.4. Illustration of no feasible solution of the constrained optimization problem. Solid-line $x_1 + x_2 = 1$; darker-solid-line $2x_1 + 2x_2 = 6$

In summary, in order to find the optimal constrained solution, the linear equality constraints are required to be linearly independent.

Example 2.7. In this example, we will show how the number of equality constraints is also an issue in the constrained optimal solution.

Once again, we use the same objective function as in Example 2.5 where $E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$; $F = \begin{bmatrix} -2 \\ -3 \\ -1 \end{bmatrix}$, but add an extra constraint to the original constraints so that

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ 3x_1 - 2x_2 - 3x_3 &= 1 \\ x_1 - 3x_2 + 2x_3 &= 1. \end{aligned} \tag{2.38}$$

Solution. The additional constraint is independent of the first two original constraints. Now the only feasible solution that satisfies all the constraints is $x = M^{-1}\gamma = [0.6552 \ 0.069 \ 0.2759]^T$, which is the unique solution of the linear equations (2.38). There is no point in proceeding to the optimization of the objective function, because the only feasible solution is the constrained optimal solution.

In summary, the number of equality constraints is required to be less than or equal to the number of decision variables (*i.e.*, x). If the number of equality constraints equals the number of decision variables, the only feasible solution is the one that satisfies the constraints and there is no additional variable in x that can be used to optimize the original objective function. If the number of equality constraints is greater than the number of decision variables,

then there is no feasible solution to satisfy the constraints. Alternatively, the situation is called infeasible.

2.4.2 Minimization with Inequality Constraints

In the minimization with inequality constraints, the number of constraints could be larger than the number of decision variables. The inequality constraints $Mx \leq \gamma$ as in (2.28) may comprise active constraints and inactive constraints. An inequality $M_i x \leq \gamma_i$ is said to be active if $M_i x = \gamma_i$ and inactive if $M_i x < \gamma_i$, where M_i together with γ_i form the i th inequality constraint and are the i th row of M matrix and the i th element of γ vector, respectively. We introduce the Kuhn-Tucker conditions, which define the active and inactive constraints in terms of the Lagrange multipliers.

Kuhn-Tucker Conditions

The necessary conditions for this optimization problem (Kuhn-Tucker conditions) are

$$\begin{aligned} Ex + F + M^T \lambda &= 0 \\ Mx - \gamma &\leq 0 \\ \lambda^T (Mx - \gamma) &= 0 \\ \lambda &\geq 0, \end{aligned} \tag{2.39}$$

where the vector λ contains the Lagrange multipliers. These conditions can be expressed in a simpler form in terms of the set of active constraints. Let S_{act} denote the index set of active constraints. Then the necessary conditions become

$$\begin{aligned} Ex + F + \sum_{i \in S_{act}} \lambda_i M_i^T &= 0 \\ M_i x - \gamma_i &= 0 \quad i \in S_{act} \end{aligned} \tag{2.40}$$

$$M_i x - \gamma_i < 0 \quad i \notin S_{act} \tag{2.41}$$

$$\lambda_i \geq 0 \quad i \in S_{act} \tag{2.42}$$

$$\lambda_i = 0 \quad i \notin S_{act}, \tag{2.43}$$

where M_i is the i th row of the M matrix. In other words, (2.40) says that for the i th row, $M_i x - \gamma_i = 0$ means that this is an equality constraint, hence an active constraint. In contrast, $M_i x - \gamma_i < 0$ (see (2.41)) means that the constraint is satisfied, hence it is an inactive constraint. For an active constraint, the corresponding Lagrange multiplier is non-negative (see (2.42)), whilst the Lagrange multiplier is zero if the constraint is inactive (see (2.43)).

It is clear that if the active set were known, the original problem could be replaced by the corresponding problem having equality constraints only.

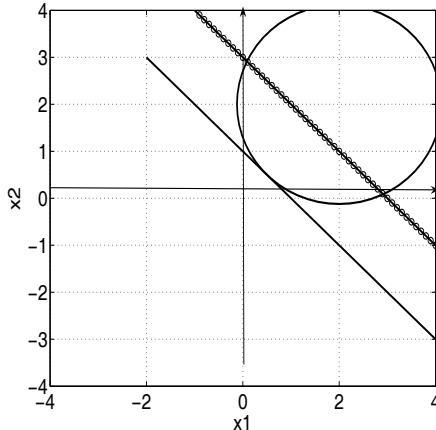


Fig. 2.5. Illustration of the constrained optimization problem with inequality constraints. Solid-line $x_1 + x_2 = 1$; darker-solid-line $3x_1 + 3x_2 = 6$

Explicitly, supposing that M_{act} and λ_{act} are given, the optimal solution with inequality solution has the closed-form

$$\lambda_{act} = -(M_{act}E^{-1}M_{act}^T)^{-1}(\gamma_{act} + M_{act}E^{-1}F) \quad (2.44)$$

$$x = -E^{-1}(F + M_{act}^T\lambda_{act}). \quad (2.45)$$

Example 2.8. In Example 2.6, we showed that when optimizing with equality constraints, if the constraints are dependent, then there is no feasible solution. In this example, we examine what happens to the optimal solution if they are inequality constraints. We assume that the objective function

$$J = \frac{1}{2}x^TEx + x^TF, \quad (2.46)$$

where the matrices $E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $F = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$ and the constraints are

$$x_1 + x_2 \leq 1 \quad (2.47)$$

$$2x_1 + 2x_2 \leq 6. \quad (2.48)$$

Solution. Clearly the set of variables that satisfy inequality (2.47) will also satisfy the inequality (2.48). This is illustrated in Figure 2.5. Thus, the constraint (2.47) is an active constraint, while the constraint (2.48) is an inactive constraint. We find the constrained optimum by minimizing J subject to equality constraint: $x_1 + x_2 = 1$, which is $x_1 = 0.5$ and $x_2 = 0.5$. We verify that indeed with this set of x_1 and x_2 values, inequality (2.48) is satisfied.

Active Set Methods

The idea of active set methods is to define at each step of an algorithm a set of constraints, termed the working set, that is to be treated as the active set. The working set is chosen to be a subset of the constraints that are actually active at the current point, and hence the current point is feasible for the working set. The algorithm then proceeds to move on the surface defined by the working set of constraints to an improved point. At each step of the active set method, an equality constraint problem is solved. If all the Lagrange multipliers $\lambda_i \geq 0$, then the point is a local solution to the original problem. If, on the other hand, there exists a $\lambda_i < 0$, then the objective function value can be decreased by relaxing the constraint i (*i.e.*, deleting it from the constraint equation). During the course of minimization, it is necessary to monitor the values of the other constraints to be sure that they are not violated, since all points defined by the algorithm must be feasible. It often happens that while moving on the working surface, a new constraint boundary is encountered. It is necessary to add this constraint to the working set, then proceed to the re-defined working surface. To illustrate the basic idea of active set methods, let us look at the example below.

Example 2.9. Optimize the objective function where

$$E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; F = \begin{bmatrix} -2 \\ -3 \\ -1 \end{bmatrix},$$

subject to the inequality constraints:

$$\begin{aligned} x_1 + x_2 + x_3 &\leq 1 \\ 3x_1 - 2x_2 - 3x_3 &\leq 1 \\ x_1 - 3x_2 + 2x_3 &\leq 1. \end{aligned} \tag{2.49}$$

Solution. The feasible solution of equality constraints (2.49) exists, which is the solution of the linear equations

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ 3x_1 - 2x_2 - 3x_3 &= 1 \\ x_1 - 3x_2 + 2x_3 &= 1. \end{aligned} \tag{2.50}$$

Thus, the three equality constraints are taken as the first working set. We calculate the Lagrange multiplier for the three constraints leading to

$$\lambda = -(ME^{-1}M^T)^{-1}(\gamma + ME^{-1}F) = \begin{bmatrix} 1.6873 \\ 0.0309 \\ -0.4352 \end{bmatrix}.$$

Clearly the third element in λ is negative, therefore, the third constraint is an inactive constraint and will be dropped from the constrained equation

set. We take the first two constraints as the active constraints, and solve the optimization problem as minimizing

$$J = \frac{1}{2}x^TEx + x^TF,$$

subject to

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ 3x_1 - 2x_2 - 3x_3 &= 1, \end{aligned} \tag{2.51}$$

which is the identical problem to the one given in Example 2.5. We solve this equality constraint problem, obtaining as before

$$\lambda = \begin{bmatrix} 1.6452 \\ -0.0323 \end{bmatrix}.$$

Clearly the second element in λ is negative. We drop the second constraint and solve the optimization problem as

$$J = \frac{1}{2}x^TEx + x^TF,$$

subject to

$$x_1 + x_2 + x_3 = 1.$$

Once more, we solve this equality constrained optimization problem, and obtain $\lambda = \frac{5}{3}$, leading to $x = [0.3333 \ 1.3333 \ -0.6667]^T$. Clearly, the optimal solution x satisfies the equality constraint. We also check whether the rest of the inequality constraints (2.49) are satisfied. They are all indeed satisfied.

There are several comments as follows.

1. In the case of equality constraints, the maximum number of equality constraints equals the number of decision variables. In this example, it is 3, and the only feasible solution x is to satisfy the equality constraints (see (2.50)). In contrast, in the case of inequality constraints, the number of inequality constraints is permitted to be larger than the number of decision variables, as long as they are not all active. In this example, only one constraint becomes active so it becomes an equality constraint. Once the optimal solution is found against this active constraint, the rest of the inequalities are automatically satisfied.
2. It is clear that an iterative procedure is required to solve the optimization problem with inequality constraints, because we did not know which constraints would become active constraints. If the active set could be identified in advance, then the iterative procedure would be shortened.
3. Note that the conditions for the inequality constraints are more relaxed than the case of imposing equality constraints. For instance, the number of constraints is permitted to be greater than the number of decision

variables, and the set of inequality constraints is permitted to be linearly dependent. However, these relaxations are only permitted to the point that the active constraints need to be linearly independent and the number of active constraints needs to be less than or equal to the number of decision variables.

2.4.3 Primal-Dual Method

The family of active methods belongs to the group of primal methods, where the solutions are based on the decision variables (also called primal variables in the literature). In the active set methods, the active constraints need to be identified along with the optimal decision variables. If there are many constraints, the computational load is quite large. Also, the programming of an active method is not a straightforward task, as we illustrated through Example 2.9. A dual method can be used systematically to identify the constraints that are not active. They can then be eliminated in the solution. The Lagrange multipliers are called dual variables in the optimization literature. This method will lead to very simple programming procedures for finding optimal solutions of constrained minimization problems.

The dual problem to the original primal problem is derived as follows. Assuming feasibility (*i.e.*, there is an x such that $Mx < \gamma$), the primal problem is equivalent to

$$\max_{\lambda \geq 0} \min_x [\frac{1}{2}x^T Ex + x^T F + \lambda^T (Mx - \gamma)]. \quad (2.52)$$

The minimization over x is unconstrained and is attained by

$$x = -E^{-1}(F + M^T \lambda). \quad (2.53)$$

Substituting this in (2.52), the dual problem is written as

$$\max_{\lambda \geq 0} \left(-\frac{1}{2}\lambda^T H\lambda - \lambda^T K - \frac{1}{2}F^T E^{-1}F \right), \quad (2.54)$$

where the matrices H and K are given by

$$H = ME^{-1}M^T \quad (2.55)$$

$$K = \gamma + ME^{-1}F. \quad (2.56)$$

Thus, the dual is also a quadratic programming problem with λ as the decision variable. Equation (2.54) is equivalent to

$$\min_{\lambda \geq 0} \left(\frac{1}{2}\lambda^T H\lambda + \lambda^T K + \frac{1}{2}\gamma^T E^{-1}\gamma \right). \quad (2.57)$$

Note that the dual problem may be much easier to solve than the primal problem because the constraints are simpler (see Hildreth's quadratic programming procedure in the next section).

The set of optimal Lagrange multipliers that minimize the dual objective function

$$J = \frac{1}{2} \lambda^T H \lambda + \lambda^T K + \frac{1}{2} \gamma^T E^{-1} \gamma, \quad (2.58)$$

subject to $\lambda \geq 0$, are denoted as λ_{act} , and the corresponding constraints are described by M_{act} and γ_{act} . With the values of λ_{act} and M_{act} , the primal variable vector x is obtained using

$$x = -E^{-1}F - E^{-1}M_{act}^T \lambda_{act}, \quad (2.59)$$

where the constraints are treated as equality constraints in the computation.

2.4.4 Hildreth's Quadratic Programming Procedure

A simple algorithm, called Hildreth's quadratic programming procedure (Luenberger, 1969, Wismer and Chattergy, 1978), was proposed for solving this dual problem. In this algorithm, the direction vectors were selected to be equal to the basis vectors $e_i = [0 \ 0 \ \dots \ 1 \ \dots \ 0 \ 0]^T$. Then, the λ vector can be varied one component at a time. At a given step in the process, having obtained a vector $\lambda \geq 0$, we fix our attention on a single component λ_i . The objective function may be regarded as a quadratic function in this single component. We adjust λ_i to minimize the objective function. If that requires $\lambda_i < 0$, we set $\lambda_i = 0$. In either case, the objective function is decreased. Then, we consider the next component λ_{i+1} . If we consider one complete cycle through the components to be one iteration taking the vector λ^m to λ^{m+1} , the method can be expressed explicitly as

$$\lambda_i^{m+1} = \max(0, w_i^{m+1}), \quad (2.60)$$

with

$$w_i^{m+1} = -\frac{1}{h_{ii}}[k_i + \sum_{j=1}^{i-1} h_{ij} \lambda_j^{m+1} + \sum_{j=i+1}^n h_{ij} \lambda_j^m], \quad (2.61)$$

where the scalar h_{ij} is the ij th element in the matrix $H = ME^{-1}M^T$, and k_i is the i th element in the vector $K = \gamma + ME^{-1}F$. Also note that in (2.61) there are two sets of λ values in the computation: one involves λ^m and one involves the updated λ^{m+1} .

Because the converged λ^* vector contains either zero or positive values of the Lagrange multipliers, by (2.53), we have

$$x = -E^{-1}(F + M^T \lambda^*). \quad (2.62)$$

There are a few comments to be made. First, Hildreth's quadratic programming algorithm is based on an element-by-element search, therefore, it does not require any matrix inversion. As a result, if the active constraints are linearly independent and their number is less than or equal to the number

of decision variables, then the dual variables will converge. However, if one or both of these requirements are violated, then the dual variables will not converge to a set of fixed values. The iteration will terminate when the iterative counter reaches its maximum value. Because there is no matrix inversion, the computation will continue without interruption. As we will observe in the coming examples, the algorithm will give a compromised, near-optimal solution with constraints if the situation of conflict constraints arises. This is one of the key strengths of using this approach in real-time applications, because the algorithm's ability to automatically recover from an ill-conditioned constrained problem is paramount for the safety of plant operation.

When the conditions are satisfied, the one-dimensional search technique in Hildreth's quadratic programming procedure has been shown to converge to the set of λ^* , where λ^* contains zeros for inactive constraints and the positive components corresponding to the active constraints. The positive component collected as a vector is called λ_{act}^* with its value defined by

$$\lambda_{act}^* = -(M_{act}E^{-1}M_{act}^T)^{-1}(\gamma_{act} + M_{act}E^{-1}F), \quad (2.63)$$

where M_{act} and γ_{act} are the constraint data matrix and vector with the deletion of the row elements that correspond to the zero elements in λ^* . The proof of the convergence relies on the existence of a set of bounded λ_{act}^* . This is virtually determined by the existence of the $(M_{act}E^{-1}M_{act}^T)^{-1}$ (see Wismer and Chattergy (1978)).

Example 2.10. Minimize the cost function:

$$J = \frac{1}{2}x^TEx + F^Tx,$$

where $E = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$; $F = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$. The constraints are $0 \leq x_1$, $0 \leq x_2$ and

$$3x_1 + 2x_2 \leq 4.$$

Solution. We form the linear inequality constraints

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix}. \quad (2.64)$$

The global optimal solution without constraints is

$$\begin{bmatrix} x_1^0 \\ x_2^0 \end{bmatrix} = -E^{-1}F = -\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

By substituting the global optimal solution into (2.64), we note that the inequality constraints are violated, with respect to the third constraint (*i.e.*, $3 + 2 > 4$).

To find the optimal λ^* , we form

$$H = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -1 & 0 & 3 \\ 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -5 \\ 1 & 2 & -7 \\ -5 & -7 & 29 \end{bmatrix} \quad (2.65)$$

$$K = \gamma + ME^{-1}F = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}. \quad (2.66)$$

The iteration starts at $k = 0$, with the initial conditions of $\lambda_1^0 = \lambda_2^0 = \lambda_3^0 = 0$. At $k = 1$,

$$w_1^1 + 1 = 0 \quad (2.67)$$

$$\lambda_1^1 + 2w_2^1 + 1 = 0 \quad (2.68)$$

$$-5\lambda_1^1 - 7\lambda_2^1 + 29w_3^1 - 1 = 0. \quad (2.69)$$

Solving (2.67) gives $\lambda_1^1 = \max(0, w_1^1) = 0$, solving (2.68) gives $\lambda_2^1 = \max(0, w_2^1) = 0$ and solving (2.69) gives $\lambda_3^1 = \max(0, w_3^1) = 0.0345$.

At $k = 2$,

$$w_1^2 + \lambda_2^1 - 5\lambda_3^1 + 1 = 0 \quad (2.70)$$

$$\lambda_1^2 + 2w_2^2 - 7\lambda_3^1 + 1 = 0 \quad (2.71)$$

$$-5\lambda_1^2 - 7\lambda_2^2 + 29w_3^2 - 1 = 0. \quad (2.72)$$

This gives $\lambda_1^2 = \max(0, w_1^2) = 0$, $\lambda_2^2 = \max(0, w_2^2) = 0$ and $\lambda_3^2 = \max(0, w_3^2) = 0.0345$. Since $\lambda^2 = \lambda^1$, the iterative procedure has converged. The optimal solution of λ is $\lambda_1^* = 0$, $\lambda_2^* = 0$ and $\lambda_3^* = 0.0345$. The optimal solution of x is given by

$$x^* = \begin{bmatrix} x_1^0 \\ x_2^0 \end{bmatrix} - E^{-1}M^T\lambda^* = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.1724 \\ 0.2414 \end{bmatrix} = \begin{bmatrix} 0.8276 \\ 0.7586 \end{bmatrix}. \quad (2.73)$$

From (2.73), it is seen that the constrained optimal solution consists of two parts. One is identical to the global optimal solution, and the second part is a correction term due to the active constraint.

Example 2.11. Solve a quadratic programming problem where the constraints are defined by $0 \leq x_1 \leq 1$ and $0 \leq x_2 \leq 1$ and the objective function is defined by

$$J = \frac{1}{2}[(x_1 - 2)^2 + (x_2 - 2)^2]. \quad (2.74)$$

Solution. The inequalities can be written as

$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}. \quad (2.75)$$

The objective function can be written as

$$J = \frac{1}{2}x^T Ex + F^T x, \quad (2.76)$$

where $E = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $F = [-2 -2]^T$. The global optimal solution is

$$\begin{bmatrix} x_1^0 \\ x_2^0 \end{bmatrix} = -E^{-1}F = \begin{bmatrix} 2 \\ 2 \end{bmatrix}. \quad (2.77)$$

The upper limits for x_1 and x_2 are violated. To follow Hildreth's quadratic programming procedure, we define

$$\begin{aligned} H &= ME^{-1}M^T = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \end{aligned} \quad (2.78)$$

$$K = \gamma + ME^{-1}F \quad (2.79)$$

$$= \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -2 \\ -2 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ -1 \\ 2 \end{bmatrix}. \quad (2.80)$$

We start the one-dimensional search with $\lambda_1^0 = \lambda_2^0 = \lambda_3^0 = \lambda_4^0 = 0$.

At $k = 1$,

$$w_1^1 - 1 = 0 \quad (2.81)$$

$$-\lambda_1^1 + w_2^1 + 2 = 0 \quad (2.82)$$

$$w_3^1 - 1 = 0 \quad (2.83)$$

$$-\lambda_3^1 + w_4^1 + 2 = 0. \quad (2.84)$$

This gives $\lambda_1^1 = \max(0, w_1^1) = 1$, $\lambda_2^1 = \max(0, w_2^1) = 0$, $\lambda_3^1 = \max(0, w_3^1) = 1$, $\lambda_4^1 = \max(0, w_4^1) = 0$.

At $k = 2$,

$$w_1^2 - \lambda_2^1 - 1 = 0 \quad (2.85)$$

$$-\lambda_1^2 + w_2^2 + 2 = 0 \quad (2.86)$$

$$w_3^2 - \lambda_4^1 - 1 = 0 \quad (2.87)$$

$$-\lambda_3^2 + w_4^2 + 2 = 0. \quad (2.88)$$

Similarly, solving (2.85) gives $\lambda_1^2 = \max(0, w_1^2) = 1$, solving (2.86) gives $\lambda_2^2 = \max(0, w_2^2) = 0$, solving (2.87) gives $\lambda_3^2 = \max(0, w_3^2) = 1$, and $\lambda_4^2 = \max(0, w_4^2) = 0$. Since $\lambda^2 = \lambda^1$, the iterative procedure has converged. The optimal solution of λ is $\lambda_1^* = 1$, $\lambda_2^* = 0$, $\lambda_3^* = 1$ and $\lambda_4^* = 0$.

We delete the inactive constraints and find that the constrained optimal solution is

$$x^* = x^0 - E^{-1} \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \quad (2.89)$$

As we can see, the constrained optimal solution is the solution that consists of an original global optimal solution and a correction term due to the active constraints.

2.4.5 MATLAB Tutorial: Hildreth's Quadratic Programming

Tutorial 2.1. *The objective of this tutorial is to demonstrate how to solve the constrained optimization problem using Hildreth programming. The problem is written as minimizing*

$$J = \frac{1}{2} \eta^T H \eta + \eta^T f, \quad (2.90)$$

subject to constraints

$$A_{\text{cons}} \eta \leq b. \quad (2.91)$$

Step by Step

1. Create a new file called *QPhild.m*.
2. The program finds the global optimal solution and checks if all the constraints are satisfied. If so, the program returns the optimal solution η . If not, the program then begins to calculate the dual variable λ .
3. Enter the following program into the file:

```
function eta=QPhild(H,f,A_cons,b);
% E=H;
% F=f;
% M=A_cons;
% gamma=b;
% eta =x
[n1,m1]=size(A_cons);
eta=-H\f;
kk=0;
for i=1:n1
if (A_cons(i,:)*eta>b(i)) kk=kk+1;
else
kk=kk+0;
end
end
if (kk==0) return; end
```

4. Note that in the quadratic programming procedure, the i th Lagrange multiplier λ_i becomes zero if the corresponding constraint is not active. Otherwise it is positive. We need to calculate the Lagrange multipliers iteratively. We will first set-up the matrices of the dual quadratic programming, followed by the computation of the Lagrange multipliers.

5. Continue entering the following program into the file:

```
P=A_cons*(H\A_cons');
d=(A_cons*(H\f)+b);
[n,m]=size(d);
x_ini=zeros(n,m);
lambda=x_ini;
al=10;
for km=1:38
%find the elements in the solution vector one by one
% km could be larger if the Lagranger multiplier has a slow
% convergence rate.
lambda_p=lambda;
for i=1:n
w= P(i,:)*lambda-P(i,i)*lambda(i,1);
w=w+d(i,1);
la=-w/P(i,i);
lambda(i,1)=max(0,la);
end
al=(lambda-lambda_p)'*(lambda-lambda_p);
if (al<10e-8); break; end
end
```

6. We can directly use the λ vector and the constraint equation to calculate the changes in η due to the active constraints, because the elements in λ are either positive or zero.

7. Continue entering the following program into the file:

```
eta=-H\f -H\A_cons'*lambda;
```

8. Test your program using the data matrices generated from Examples 2.10 and 2.11. If the constrained optimal solutions are identical to the solutions given in the examples, then your program is correct.

2.4.6 Closed-form Solution of λ^*

We noticed that the Hildreth quadratic programming procedure produces the optimal λ^* that has zeros and the components corresponding to the active constraints. The converged vector is called λ_{act}^* with its value defined by

$$\lambda_{act}^* = -(M_{act}E^{-1}M_{act}^T)^{-1}(\gamma_{act} + M_{act}E^{-1}F), \quad (2.92)$$

where M_{act} and γ_{act} are the constraint data matrix and vector with the deletion of the row elements that correspond to the zero elements in λ^* . Thus, if we could correctly identify *a priori* the active constraints, then we can compute the closed-form solution of the constrained optimization problem.

Example 2.12. Suppose that with *a priori* knowledge, the third constraint in Example 2.10 is known to be active. Find the optimal solution using the closed-form solution.

Solution. With the given information, we let $\lambda_1^* = \lambda_2^* = 0$. We form

$$M_{act} = \begin{bmatrix} 3 & 2 \end{bmatrix}; \gamma_{act} = 4$$

$$\lambda_{act}^* = -(M_{act}E^{-1}M_{act}^T)^{-1}(\gamma_{act} + M_{act}E^{-1}F) = \frac{1}{29} = 0.0345.$$

The results are identical to those obtained in Example 2.10.

Example 2.13. Assume that the active constraints in Example 2.11 are number one and number three. Find the optimal solution with respect to the constraints using the closed-form solution.

Solution. With the *a priori* knowledge, $\lambda_2^* = \lambda_4^* = 0$. The active constraints are formed using number one and number three constraints as

$$M_{act} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \gamma_{act} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Hence, the optimal dual variable is solved as

$$\lambda_{act}^* = -(M_{act}E^{-1}M_{act}^T)^{-1}(\gamma_{act} + M_{act}E^{-1}F) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Because the elements in λ_{act}^* are positive, the solution is accepted as the optimal solution for λ^* . Again, the results are identical to those obtained in Example 2.11.

This technique of guessing active constraints is useful in the situation when the computational speed is critical for an application, with which we can resort to the closed-form solution of a constrained control problem. Also, the computational speed is increased if we only need to search for part of the active constraints set, while the other part comes from guesswork.

2.5 Predictive Control with Constraints on Input Variables

This section will present several worked examples of predictive control showing how constraints on the input variable $u(k)$ are imposed. The constraints include those on rate of change and amplitude constraints. These constraints are commonly encountered in industrial applications. MATLAB tutorials for constrained control can be found in Chapter 3.

2.5.1 Constraints on Rate of Change

We present two worked examples of predictive control here. The first example is to show how to incorporate constraints for the first element in ΔU , and the second example is to show how to incorporate constraints for all the elements in ΔU . The first example demands less in computation, but sometimes it can compromise the closed-loop performance of the predictive control system.

Example 2.14. A continuous-time plant is described by a transfer function model,

$$G(s) = \frac{10}{s^2 + 0.1s + 3}, \quad (2.93)$$

which has a pair of poles at $-0.0500 \pm 1.7313j$. Suppose that the system is sampled with an interval $\Delta = 0.1$. Design a discrete-time model predictive control with control horizon $N_c = 3$, $N_p = 20$, and $\bar{R} = 0.01 \times I$. The limit on the rate of change on the control signal is specified as

$$-1.5 \leq \Delta u(k) \leq 3.$$

For this example, we only consider the case of imposing the constraints on the first element of ΔU .

Solution. By following Tutorial 1.1 we first obtain the discrete-time state-space model, then augment the model with an integrator. With the program presented in Tutorial 1.2, we obtain the objective function:

$$J = \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U - 2\Delta U^T \Phi^T (R_s - Fx(k_i)), \quad (2.94)$$

where

$$\Phi^T \Phi = \begin{bmatrix} 0.1760 & 0.1553 & 0.1361 \\ 0.1553 & 0.1373 & 0.1204 \\ 0.1361 & 0.1204 & 0.1057 \end{bmatrix}; \Phi^T F = \begin{bmatrix} 0.1972 & -0.1758 & 1.4187 \\ 0.1740 & -0.1552 & 1.2220 \\ 0.1522 & -0.1359 & 1.0443 \end{bmatrix},$$

and $R_s = \begin{bmatrix} 1.4187 \\ 1.2220 \\ 1.0443 \end{bmatrix}$ $r(k_i)$. $r(k_i)$ and $\hat{x}(k_i)$ are the set-point signal and the estimated state variable at time k_i , respectively. For simplicity, we assume that the observer poles are selected at 0, 0, 0. The closed-loop system without constraints has its eigenvalues located at 0.6851, $0.9109 \pm 0.1070j$, and 0, 0, 0. For a unit set-point change, the closed-loop responses are presented in Figure 2.6.

Based on the procedures presented in Section 2.3, the constraints are translated to the two linear inequalities as

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i + 1) \\ \Delta u(k_i + 2) \end{bmatrix} \leq \begin{bmatrix} 3.0000 \\ 1.5000 \end{bmatrix}. \quad (2.95)$$

To demonstrate how the solution evolves, we illustrate the first two steps in the computational procedure.

1. Assume that the initial observer state is zero, and a set-point signal at time $k_i = 1$, is 1. Then, without constraints, the global optimal solution of ΔU is $[6.1083 \ 2.3334 \ -0.5861]^T$. Thus, the constraint (2.95) is violated. The problem is solved using Hildreth's quadratic programming to obtain

$$\Delta U = [3.0000 \ 4.2751 \ 1.0494]^T.$$

2. With the first component $\Delta u(1) = 3$, and $y(1) = 0$ the estimated state variable is updated to yield $\hat{x}(2) = [3.0000 \ 0 \ 0.0015]^T$.
3. Again, the global optimal solution is $\Delta U = [4.6329 \ 1.1265 \ -1.5559]^T$, which violates the constraint. The optimization problem is again solved using the Hildreth's quadratic programming to obtain

$$\Delta U = [3.0000 \ 2.1466 \ -0.6967]^T.$$

4. With updated information on $y(2) = 0.0015$ and $\Delta u(2) = 3$, the estimated state variable is updated with value $\hat{x}(3) = [8.9961 \ 3.0000 \ 0.0075]^T$.
5. The global optimal solution is $\Delta U = [2.9338 \ -0.2126 \ -2.5828]^T$, which satisfies the constraint.
6. The computation continues simultaneously in closed-loop simulation and development of constrained control.

Figure 2.6 shows the control signal, plant output and Δu in the presence of the constraints, where the comparison with the unconstrained solution is also illustrated. It is seen from Figure 2.6b that the constraints on Δu are satisfied, whilst Figure 2.6a shows that the control signal and output responses have very small differences in the presence of constraints.

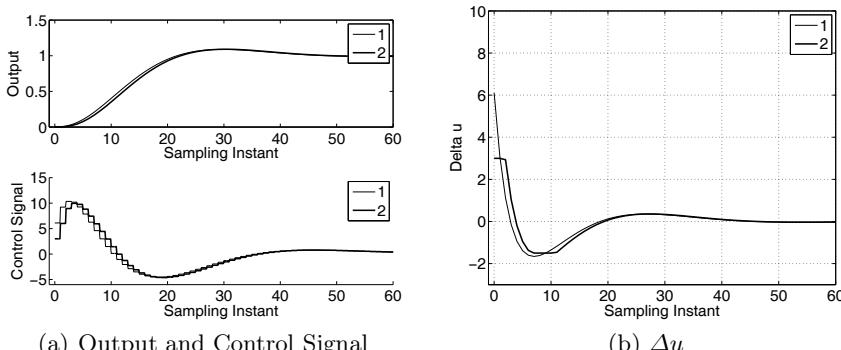


Fig. 2.6. Closed-loop system response with constraints. Key: line (1) without constraints; line (2) with constraints $-1.5 \leq \Delta u(k) \leq 3$

Example 2.15. This example will investigate the scenario where the constraints are imposed for all elements in ΔU , which is the case often referred to in the predictive control literature. The nominal design of predictive control remains the same as in Example 2.14.

Solution. From Section 2.3, when the constraints are fully imposed on all the components in ΔU , they are translated to the six linear inequalities as

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i + 1) \\ \Delta u(k_i + 2) \end{bmatrix} \leq \begin{bmatrix} 3 \\ 3 \\ 3 \\ 1.5 \\ 1.5 \\ 1.5 \end{bmatrix}. \quad (2.96)$$

Since there are three decision variables ($N_c = 3$), the number of active constraints should not be greater than 2 at any given time in order to have an optimal solution. We illustrate the solutions for the first two cycles of the computation.

1. At $k_i = 1$, without constraints, the global optimal solution is $\Delta U = [6.1083 \ 2.3334 \ -0.5861]^T$. Thus, the constraint (2.96) is violated. The problem is solved using Hildreth's quadratic programming to obtain $\Delta U = [3.0000 \ 3.0000 \ 2.3731]^T$. In comparison with the results in Example 2.14, the first solution has two active constraints. Namely, the first two elements in ΔU are the results from active constraints.
2. At $k_i = 2$, with the first component $\Delta u(1) = 3$, and $y(1) = 0$ the estimated state variable is updated to yield $\hat{x}(2) = [3.0000 \ 0 \ 0.0015]^T$.
3. Again, the global optimal solution is $\Delta U = [4.6329 \ 1.1265 \ -1.5559]^T$, which violates the constraint. The optimization problem is again solved using Hildreth's quadratic programming to obtain $\Delta U = [3.0000 \ 2.1466 \ -0.6967]^T$, where only the first constraint becomes activated, therefore the solution is identical to the solution in the second step of Example 2.14.
4. With updated information on $y(2) = 0.0015$ and $\Delta u(2) = 3$, the estimated state variable is updated with value $\hat{x}(3) = [8.9961 \ 3.0000 \ 0.0075]^T$.
5. The global optimal solution is $\Delta U = [2.9338 \ -0.2126 \ -2.5828]^T$, where the third element violates the constraint $-1.5 \leq \Delta u(k)$. The optimization problem is solved using Hildreth's quadratic programming to obtain $\Delta U = [2.4885 \ -0.6280 \ -1.5000]^T$. The effect of the third constraint becoming activated made the first component drop from 2.9338 (see the previous example) to 2.4885.

Because the constraints are imposed on all the element of ΔU , there are possibilities that non-essential constraints become activated. Figure 2.7 shows that this is indeed the case, which results in the non-smooth solution for $\Delta u(k_i)$. In Figure 2.8, we compare the results with those from Example 2.14 where the constraints were only imposed on the first sample of ΔU . The

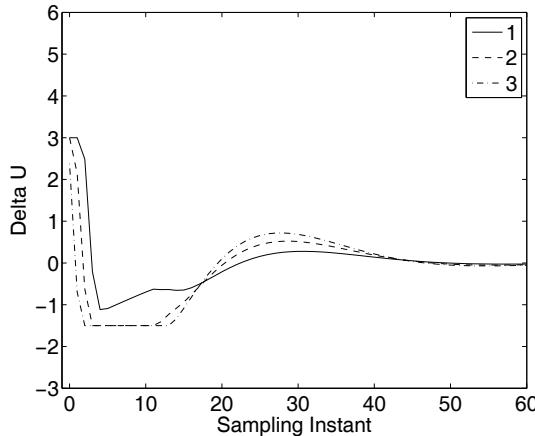


Fig. 2.7. All control elements with constraints. Key: line (1) $\Delta u(k_i)$; line (2) $\Delta u(k_i + 1)$; line (3) $\Delta u(k_i + 2)$

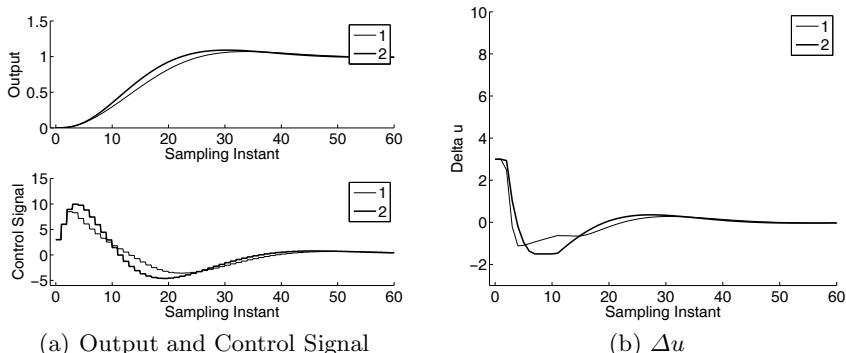


Fig. 2.8. Comparison results between Example 2.15 and 2.14. Key: line (1) solution from Example 2.15; line (2) the solution from Example 2.14

difference is caused by the constraints from the set of non-essential constraints becoming active.

2.5.2 Constraints on Amplitude of the Control

The amplitude of the control signal $u(k)$ is another important object for imposing constraint. The examples below illustrate how to impose constraints on the amplitude of the control signal. We will consider two examples. The first example is to impose the constraints on the first sample of the control signal and the second example is to impose constraints on all elements of the control signal.

Example 2.16. We will consider the same system given in Example 2.14 with identical design specification, except that the constraints are changed to

$$-3 \leq u(k) \leq 6.$$

Again, in this example, we will consider the case of imposing the constraints on the first sample of control.

Solution. Note that at sample time k_i ,

$$u(k_i) = \Delta u(k_i) + u(k_i - 1); \quad \Delta u(k_i) = [1 \ 0 \ 0] \Delta U.$$

Here, ΔU is the parameter vector to be optimized. Therefore, from Section 2.3, the inequality constraints are translated into

$$-3 \leq [1 \ 0 \ 0] \Delta U + u(k_i - 1) \leq 6.$$

That is,

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i + 1) \\ \Delta u(k_i + 2) \end{bmatrix} \leq \begin{bmatrix} 6 - u(k_i - 1) \\ 3 + u(k_i - 1) \end{bmatrix}. \quad (2.97)$$

Note that the value of past control signal $u(k_i - 1)$ is embedded into the constraint equations, thus the constraint equations need to be updated as the control system is implemented in real time.

Let us examine the first two cycles of the implementation.

1. When $k_i = 1$, without constraints, the global optimal solution is:

$$\Delta U = [6.1083 \ 2.3334 \ -0.5861]^T,$$

which gives $u(k_i) = 6.108$, where $u(k_i - 1) = 0$ is assumed as the initial condition. Therefore, the constraint is violated. The quadratic program procedure finds the optimal solution as $\Delta U = [6.0000 \ 2.4010 \ -0.5291]^T$, which leads to $u(1) = 6$ satisfying the constraint.

2. With the updated information on the estimated state variable and the information $u(1) = 6$, with constraint on the control amplitude, the optimal solution for $k_i = 2$ is $\Delta U = [0 \ 1.8921 \ -0.8643]^T$, which gives the optimal control as $u(2) = 6$ satisfying the constraint.
3. As time progresses, the predictive control system finds the optimal control at each sampling period with respect to the specified constraints on the control signal.

Figure 2.9 shows the constrained control response in comparison with the case without using constraints. It is seen that constraints are satisfied, and the output response remains close to the response obtained from the unconstrained case.

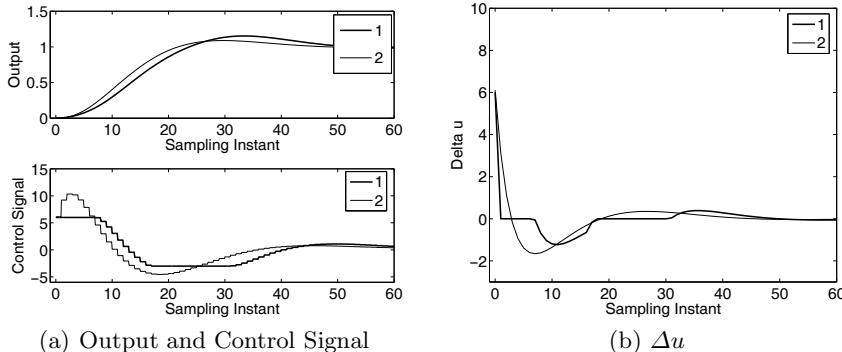


Fig. 2.9. Closed-loop response with constraints (Example 2.16). Key: line (1) solution with constraints; line (2) solution without constraints

Example 2.17. In this example, we will continue the Example 2.16 by considering the case $-3 \leq u(k) \leq 6$, however, where the constraints are imposed on all elements of the control signal.

Solution. Note that

$$u(k_i + 1) = \Delta u(k_i + 1) + u(k_i) = \Delta u(k_i + 1) + \Delta u(k_i) + u(k_i - 1);$$

$$u(k_i + 2) = \Delta u(k_i + 2) + \Delta u(k_i + 1) + \Delta u(k_i) + u(k_i - 1).$$

In matrix vector form, the expression for the three elements of control in terms of ΔU is

$$\begin{bmatrix} u(k_i) \\ u(k_i+1) \\ u(k_i+2) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i+1) \\ \Delta u(k_i+2) \end{bmatrix} + \begin{bmatrix} u(k_i-1) \\ u(k_i-1) \\ u(k_i-1) \end{bmatrix}. \quad (2.98)$$

Now, with constraints to be imposed on both upper and lower limits of the control signals, the linear inequalities are formulated into the matrix vector form

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ -1 & 0 & 0 \\ -1 & -1 & 0 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i + 1) \\ \Delta u(k_i + 2) \end{bmatrix} \leq \begin{bmatrix} 6 - u(k_i - 1) \\ 6 - u(k_i - 1) \\ 6 - u(k_i - 1) \\ 3 + u(k_i - 1) \\ 3 + u(k_i - 1) \\ 3 + u(k_i - 1) \end{bmatrix}. \quad (2.99)$$

With the same implementation as before, the predictive control finds the optimal solution that satisfies the six inequality constraints. Figure 2.10 shows the three components in the optimal solution ΔU . It is seen that the two additional constraints on $u(k_i+1)$ and $u(k_i+2)$ are activated during the process of optimization, which is indicated by the zero values of the components in

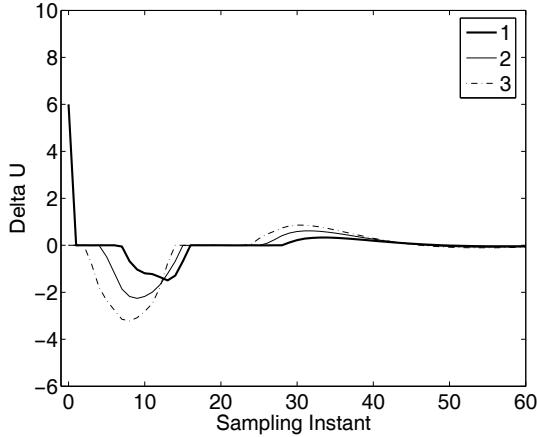


Fig. 2.10. All control elements. Key: line (1) $\Delta u(k_i)$; Line(2) $\Delta u(k_i + 1)$; line (3) $\Delta u(k_i + 2)$.

ΔU . Strictly speaking, the active constraints are indicated by the positive values of the Lagrange multipliers. However, when the amplitude of the control reaches its limit, the corresponding component in ΔU is zero so as to satisfy the constraint in the solution. In order to illustrate the difference between the approach where the constraints are imposed on all future control and the one with only constraints on $u(k_i)$, Figure 2.11 shows the plant output, control signal and $\Delta u(k)$. Again, the difference is insignificant. However, the number of constraints is three times larger when the constraints are imposed for all future control movements.

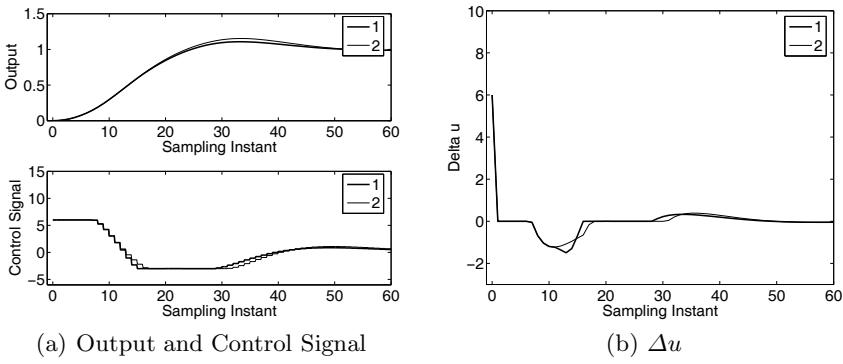


Fig. 2.11. Comparison of results between Example 2.16 and 2.17. Key: line (1) solution with constraints on all future control; line (2) solution with constraints on the first sample of future control

2.5.3 Constraints on Amplitude and Rate of Change

It is also a common practice that constraints are imposed on both the amplitude and the rate of change of control signal. If this is required in the design specification, both constraints will be combined together to form a larger set of linear inequalities. Two examples are given below to illustrate the design and implementation procedure. Again, we will consider first the case where constraints are imposed at the sampling instant k_i , and secondly extend the constraints to all future control movements.

Example 2.18. We consider the same system as in Example 2.14 with constraints on

$$-1.5 \leq \Delta u(k) \leq 3; \quad -3 \leq u(k) \leq 6.$$

Solution. Assuming that at the sampling instant k_i the constraints are only imposed on $\Delta u(k_i)$ and $u(k_i)$, the set of linear inequality constraints is formulated into the matrix vector form

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i + 1) \\ \Delta u(k_i + 2) \end{bmatrix} \leq \begin{bmatrix} 3.0000 \\ 1.5000 \\ 6 - u(k_i - 1) \\ 3 + u(k_i - 1) \end{bmatrix}. \quad (2.100)$$

The first two rows are used for the rate constraints and the last two rows are for the amplitude constraints. We solve the constrained optimization problem by minimizing the objective function J subject to the constraints given by (2.100). The control, output and $\Delta u(k)$ signals are shown in Figure 2.12. It is seen that both constraints are satisfied with a small amount of performance change when compared with the case of optimal control without constraints, which can be viewed in Figure 2.12.

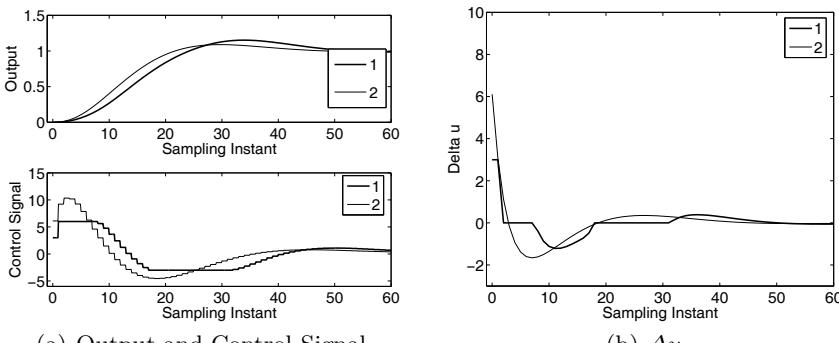


Fig. 2.12. Closed-loop control response with constraints (Example 2.18). Key: line (1) responses with constraints on both amplitude and rate of change; line (2) responses without constraints

Example 2.19. In this example, we consider the case

$$-1.5 \leq \Delta u(k) \leq 3; \quad -3 \leq u(k) \leq 6.$$

However, the constraints will be imposed on all future components of ΔU and $u(k_i), u(k_i+1), u(k_i+2)$ and the results will be compared with those presented in Example 2.18.

Solution. With all the constraints imposed on both the rate of change and the amplitude of the control signal, the inequalities are translated into the matrix vector form

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ -1 & 0 & 0 \\ -1 & -1 & 0 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i + 1) \\ \Delta u(k_i + 2) \end{bmatrix} \leq \begin{bmatrix} 3 \\ 3 \\ 3 \\ 1.5 \\ 1.5 \\ 1.5 \\ 6 - u(k_i - 1) \\ 6 - u(k_i - 1) \\ 6 - u(k_i - 1) \\ 3 + u(k_i - 1) \\ 3 + u(k_i - 1) \\ 3 + u(k_i - 1) \end{bmatrix}. \quad (2.101)$$

The predictive control system is solved by optimizing the objective function (2.94) subject to the constraints defined in (2.101). Figure 2.13 shows the optimal solution for the three components in ΔU . It is seen that all constraints are satisfied. With this predictive control, the plant control, output and $\Delta u(k)$ signals are shown in Figure 2.14. In order to see the improvement of this case over the previous case in Example 2.18, the three signals are compared with those obtained when the constraints on the first sample are imposed.

There is some improvement in the performance with a slightly smaller over-shoot in the output response. However, the difference is very small, as viewed in Figure 2.14.

2.5.4 Constraints on the Output Variable

In this section, we investigate the case where the constraints are imposed on the output variable. Here, we will use a state-space model that has the state variables measurable.

Example 2.20. Assume that a discrete-time system is described by the z-transfer function

$$\frac{Y(z)}{U(z)} = \frac{0.0048z + 0.0047}{(z - 1)(z - 0.9048)},$$

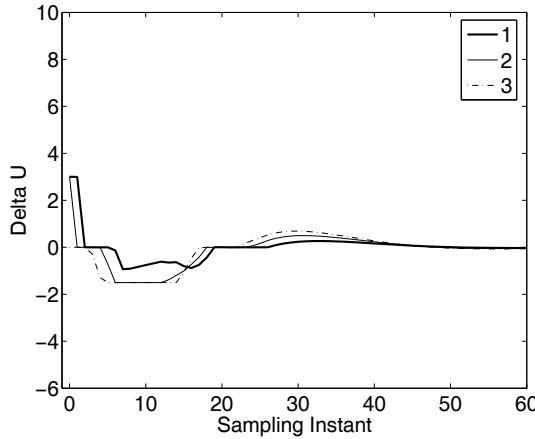


Fig. 2.13. All control elements with constraints. Key: line (1) $\Delta u(k_i)$; line (2) $\Delta u(k_i + 1)$; line (3) $\Delta u(k_i + 2)$

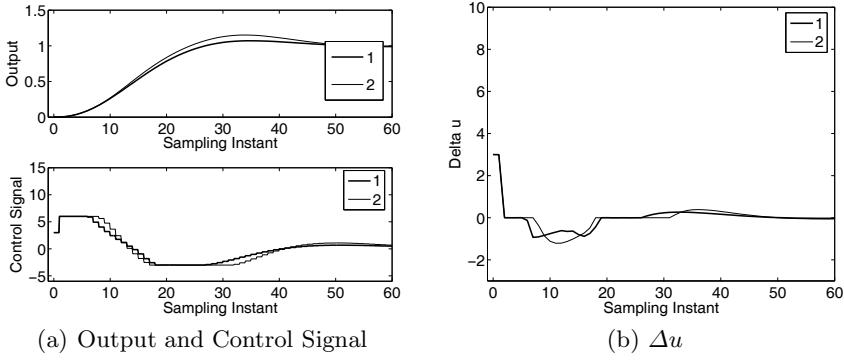


Fig. 2.14. Comparison of results between Examples 2.18 and 2.19. Key: line (1) solution with constraints on all future control; line (2) solution with constraints on the first sample of future control

which corresponds to the equation with shift operator q ,

$$(q^2 - 1.9048q + 0.9048)y(k) = (0.0048q + 0.0047)u(k),$$

where $qy(k) = y(k+1)$ and $qu(k) = u(k+1)$. Choosing the state variable $x_m(k) = [y(k) \ y(k-1) \ u(k-1)]^T$, the state-space model with this set of state variables is given by

$$\begin{bmatrix} y(k+1) \\ y(k) \\ u(k) \end{bmatrix} = \begin{bmatrix} 1.9048 & -0.9048 & 0.0047 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y(k) \\ y(k-1) \\ u(k-1) \end{bmatrix} + \begin{bmatrix} 0.0048 \\ 0 \\ 1 \end{bmatrix} u(k)$$

$$y(k) = [1 \ 0 \ 0] x_m(k). \quad (2.102)$$

Because the variables in $x_m(k)$ are measurable, there is no need to use an observer in the implementation of the predictive control system.

Assume that $N_p = 46$, $N_c = 8$, and $\bar{R} = 0.1 \times I$. Here, with zero initial conditions of the state variables, a unit step set-point change occurs at sample time $k = 0$. Then at $k = 20$, an input unit step disturbance is introduced into the system and at $k = 40$, a negative unit input disturbance is introduced. The operating constraint is that

$$0 \leq y(k) \leq 1.$$

Design and simulate the predictive control system with output constraints. Also, examine the case when the following input constraints are present,

$$-1.2 \leq u(k) \leq 1.8, \quad -0.5 \leq \Delta u(k) \leq 0.5.$$

For simplicity, all constraints are imposed on the first sample of the signals in the optimization window.

Solution. With the performance specification, we design a predictive control system with constraints. Figure 2.15 shows the closed-loop response of the constrained control system. It is seen from the plots that the constraints on the output are satisfied. At sampling time $k = 13$ and $k = 23$, the constraints are active, where we notice two separate sharp drops occurring in the control. The first is due to the slight over-shoot in the set-point change, and the second is due to the input disturbance. By comparing the control signals with and without constraints, we also notice that there are sharp changes on the $\Delta u(k)$ as well as on the control signal $u(k)$ in order to satisfy the constraints on the output. These two sharp changes on both control and increment of control at the same time instant could cause violation of constraints if constraints on the control signal are imposed. We illustrate this by continuing this study with additional constraints on the control.

Now, suppose that there are constraints on the input variables as well as the output variable. With the identical simulation conditions, the constraints on the input variables and output variable are

$$-1.2 \leq u(k) \leq 1.8, \quad -0.5 \leq \Delta u(k) \leq 0.5, \quad 0 \leq y(k) \leq 1.$$

Figure 2.16 shows the closed-loop predictive control responses in the presence of the constraints. It is seen from the plots that the output constraints are satisfied. However, the constraints on both the amplitude and increment of the control are violated when the sharp adjustment of control is generated in order to satisfy the constraint on the output. The active constraints on input and output at the same sampling instant become linearly dependent. Therefore, something has to give. Here, without any interfering, Hildreth's programming algorithm chose a solution that satisfies the output constraint and relaxed the input constraints.

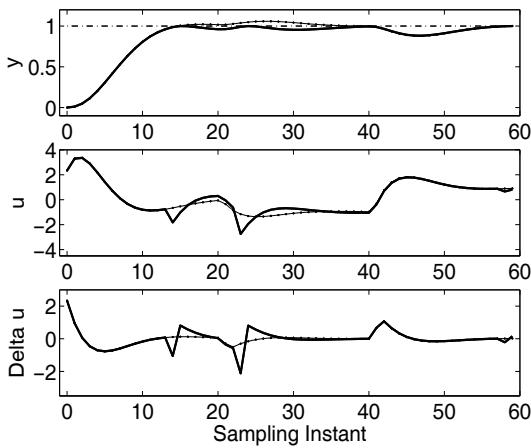


Fig. 2.15. Predictive control with output constraints $0 \leq y(k) \leq 1$. Solid line constrained response; solid dotted line unconstrained response

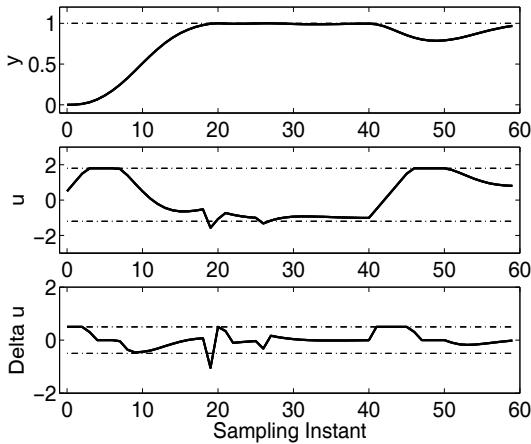


Fig. 2.16. Predictive control with constraints. $-1.2 \leq u(k) \leq 1.8$, $-0.5 \leq \Delta u(k) \leq 0.5$, $0 \leq y(k) \leq 1$

2.6 Summary

This chapter has discussed discrete-time model predictive control with constraints. Imposing constraints in the design and implementation of predictive control system involves the following steps:

1. Defining plant operational limits, including limits on the input variables, the incremental change of the input variables, state variables and plant output variables.

2. Expressing these limits as parameters for the minimum and maximum of u , Δu , x_m and y , with consideration of steady-state information.
3. With parameterization of the future control trajectory, these minimum and maximum values are expressed in the form of inequalities with $\Delta u(k_i)$, $\Delta u(k_i + 1)$, ..., $\Delta u(k_i + N_c - 1)$ as the variables.
4. The design objective of model predictive control becomes the minimization of the original error function subject to the inequality constraints, where the set of parameters $\Delta u(k_i)$, $\Delta u(k_i + 1)$, ..., $\Delta u(k_i + N_c - 1)$ become decision variables.
5. Solving the constrained optimization problem using a quadratic programming procedure at every sampling instance to obtain the optimal solution of the decision variables.

Because the constraints are expressed in terms of inequalities (constraints may or may not be violated at a particular time), in general there is no closed-form solution of the constrained control problem, unless the set of active constraints are known. If the active constraints are known, the optimal solution of the decision variables is expressed in a closed-form. In this chapter, instead of solving the decision variables iteratively, Hildreth's programming procedure was used to identify the active constraints via Lagrange multipliers (or the dual variables). Although it is still a quadratic programming problem on the dual variables, the constraints are much simplified ($\lambda \geq 0$) so a simple iterative procedure was used to obtain the optimal solution of the multipliers. Perhaps even more importantly, the iterative solution does not involve matrix inversion, so in the situation of conflict constraints, the algorithm still delivers a compromised, sub-optimal solution without being numerically unstable. This is particularly important in the real-time implementation of the predictive control system.

There is a rich literature on the topic of predictive control with constraints. The primal methods have dominated the numerical solutions in the classical literature (see for example, Muske and Rawlings, 1993, Ricker, 1985, Zafriou, 1991) until more recent years specially tailored interior-point methods applicable to MPC have appeared (Rao *et al.*, 1998, Gopal and Biegler, 1998, Hansson, 2000). Some attempts have been made to find analytical solutions (see for example, Bemporad *et al.*, 1999, Seron *et al.*, 2000). A study done by Zheng (1999) has indicated that for stable systems, reducing the number of constraints on the future control movements could cause little performance deterioration. Another interesting approach to the constrained control problem was proposed by Rossiter and Kouvaritakis (1993) where it was solved iteratively using a weighted least squares type of algorithm (Lawson's algorithm). In Tsang and Clarke (1988), optimal solutions were derived for constrained GPC of SISO systems with a control horizon of 1 or 2. The essence of their approach was to take a 'guess' at the active constraints for the two special cases and apply the closed-form solution.

Problems

2.1. Assume that the set of constraints is defined by

$$x_1 \geq 0, x_2 \geq 0, x_1 + x_2 \leq 3, 2x_1 - x_2 \leq 4, x_2 \leq 2,$$

and the objective function is

$$J = \frac{1}{2}x^T Ex + x^T F,$$

where $E = \begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}$ and $F = \begin{bmatrix} -15 & -7 \end{bmatrix}$.

1. Find the unconstrained minimum of the objective function J .
2. How many constraints are violated with this global optimal solution?
3. Draw the linear inequality constraints on (x_1, x_2) plane, also mark the global optimal solution on the plane. From this plot, take a guess at the active constraints for this constrained optimization problem. Validate your guess by treating the active constraints as equality constraints.
4. If your initial guess is not correct, take another guess at the active constraints until the validation shows that you have correctly found the set of active constraints.
5. What have you learned from this exercise?

2.2. Continue from Problem 2.1. Find the constrained minimum of J by using Hildreth quadratic programming method and compare with the answer you obtained from Problem 2.1.

2.3. Consider the discrete-time DC motor model

$$\begin{aligned} \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} &= \begin{bmatrix} 0.9048 & 0 \\ 0.0952 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.0952 \\ 0.0048 \end{bmatrix} u(k) \\ y(k) &= [0 \ 1] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}. \end{aligned} \quad (2.103)$$

Assuming the measurement of shaft position, design a predictive control system that will follow a unit set-point signal with constraints. The closed-loop observer system has poles at 0.1, 0.2 and 0.3, and the control signal satisfies the constraints

$$0 \leq u(k) \leq 0.6; \quad -0.2 \leq \Delta u(k) \leq 0.2.$$

The prediction horizon $N_p = 60$, and control horizon $N_c = 5$; $\bar{R} = I$. The initial conditions of the plant state variable vector and the observer states are assumed to be zero.

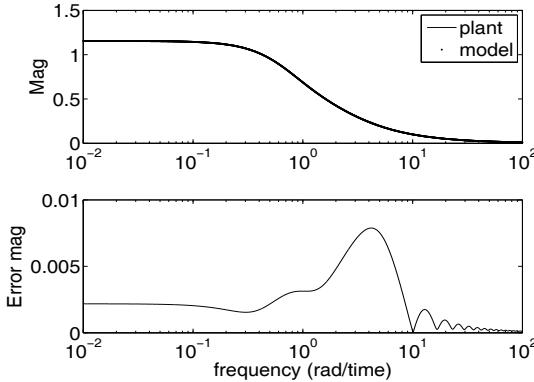


Fig. 2.17. The approximation of a recycle process; comparison of frequency responses

2.4. Transportation of materials in a feedback manner is often called plant with recycle characteristics, which is essentially embedded with time delays in the continuous-time transfer function. An irrational continuous-time transfer function used to describe such a process is given by

$$G_p(s) = \frac{1}{s + 1 - e^{-s-2}}. \quad (2.104)$$

In order to design a feedback control system for this process, one of the approaches is to approximate the continuous-time transfer function with a rational one. Such a continuous-time transfer function approximation (Wang and Cluett, 2000) is found by using a Laguerre model (see Figure 2.17), and has the form

$$G(s) = \frac{1.0117s^2 + 2.1709s + 1.4949}{(s + 1.09)^3}.$$

Assume that the control objective of the predictive control system is to maintain the output to be constant while rejecting input step disturbance. Choosing the sampling interval $\Delta t = 0.2$, $\bar{R} = I$, $N_c = 4$, $N_p = 20$, design a discrete-time model predictive control system with constraints, assuming a unit step set-point signal and zero initial conditions on the state variables, where the constraints are specified as

$$-0.1 \leq \Delta u(k) \leq 0.1; \quad 0 \leq u(k) \leq 1.1.$$

An observer may be used in the control system, where the closed-loop poles of the observer are chosen to be 0.1, 0.2, 0.3 and 0.4. The constraints are imposed on the first two samples of the variables. Simulate the nominal closed-loop performance by assuming zero initial conditions of u and y . In this simulation, the plant is assumed to be identical to the approximate model.

Discrete-time MPC Using Laguerre Functions

3.1 Introduction

In essence, the core technique in the design of discrete-time MPC is based on optimizing the future control trajectory, that is the difference of the control signal, $\Delta u(k)$. By assuming a finite control horizon N_c , the difference of the control signal $\Delta u(k)$ for $k = 0, 1, 2, \dots, N_c - 1$ is captured by the control vector ΔU while the rest of the $\Delta u(k)$ for $k = N_c, N_c + 1, \dots, N_p$ is assumed to be zero. In the examples encountered before, there were cases where the neglected trajectory $\Delta u(k)$ was not zero, however, it was small in its magnitude. The idea in this chapter is to generalize the design procedure by introducing a set of discrete orthonormal basis functions into the design. For the present, let us focus on the basic ideas by treating it as a generalization of the basic approach. This generalization will help us in reformulating the predictive control problem and simplifying the solutions, in addition to tuning the predictive control system. Furthermore, a long control horizon can be realized without using a large number of parameters. Several MATLAB tutorials are presented in this chapter for the design of discrete-time predictive control systems, with or without constraints.

3.2 Laguerre Functions and DMPC

For notational simplicity the single-input and single-output case is examined first, and the results are extended to multi-input and multi-output cases later in the chapter.

Recall that the control vector that is optimized in the design of predictive control is ΔU , defined by

$$\Delta U = [\Delta u(k_i) \Delta u(k_i + 1) \dots \Delta u(k_i + N_c - 1)]^T,$$

where the dimension of the control vector is N_c , called the control horizon. At time k_i , any element within ΔU can be represented using the discrete

δ -function in conjunction with ΔU

$$\Delta u(k_i + i) = [\delta(i) \delta(i-1) \dots \delta(i-N_c+1)] \Delta U,$$

where $\delta(i) = 1$, if $i = 0$; and $\delta(i) = 0$ if $i \neq 0$. The δ function acts like a pulse (and is also called the pulse operator), and the function $\delta(i-d)$ shifts the centre of the pulse forward as the index d increases. From this expression, it is clear that pulse operators are used in capturing the control trajectory if we regard ΔU as the coefficient vector. It is understood that $\Delta u(k_i + i)$, $i = 0, 1, \dots, N_c - 1$ can be approximated by a discrete polynomial function. There are many approaches to using discrete polynomial functions. What we propose here is to use a set of discrete Laguerre functions to approximate the sequence $\Delta u(k_i)$, $\Delta u(k_i + 1)$, ..., $\Delta u(k_i + N_c - 1)$. For the time being, let us introduce discrete-time Laguerre functions, and the reasons for using this set of functions will be justified later.

3.2.1 Discrete-time Laguerre Networks

The discrete-time Laguerre network was generated from the discretization of a continuous-time Laguerre network (a more detailed discussion on continuous-time Laguerre functions can be found in Chapter 5). The z-transforms of the discrete-time Laguerre networks are written as

$$\begin{aligned} \Gamma_1(z) &= \frac{\sqrt{1-a^2}}{1-az^{-1}} \\ \Gamma_2(z) &= \frac{\sqrt{1-a^2}}{1-az^{-1}} \frac{z^{-1}-a}{1-az^{-1}} \\ &\vdots \\ \Gamma_N(z) &= \frac{\sqrt{1-a^2}}{1-az^{-1}} \left(\frac{z^{-1}-a}{1-az^{-1}} \right)^{N-1}, \end{aligned} \quad (3.1)$$

where a is the pole of the discrete-time Laguerre network, and $0 \leq a < 1$ for stability of the network. The free parameter, a , is required to be selected by the user; this is also called the scaling factor. The Laguerre networks are well known for their orthonormality. In the frequency domain, this orthonormality is expressed in terms of the orthonormal equations for Γ_m , $m = 1, 2, \dots$, as

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_m(e^{jw}) \Gamma_m(e^{jw})^* dw = 1 \quad (3.2)$$

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} \Gamma_m(e^{jw}) \Gamma_n(e^{jw})^* dw = 0 \quad m \neq n, \quad (3.3)$$

where A^* denotes complex conjugate of A . In the design of predictive control, we explicitly use the Laguerre functions in the time domain. The discrete-time Laguerre functions are obtained through the inverse z-transform of the

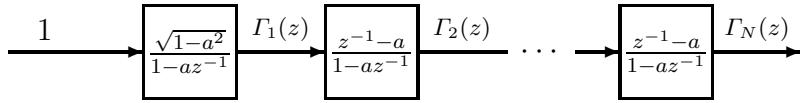


Fig. 3.1. Discrete Laguerre network

Laguerre networks. However, taking the inverse z-transform of the Laguerre networks does not lead to a compact expression of the Laguerre functions in the time-domain. A more straightforward way to find these discrete-time functions is based on a state-space realization of the networks.

Note that

$$\Gamma_k(z) = \Gamma_{k-1}(z) \frac{z^{-1} - a}{1 - az^{-1}}, \quad (3.4)$$

with $\Gamma_1 = \frac{\sqrt{1-a^2}}{1-az^{-1}}$. With this relation, the Laguerre network is illustrated in Figure 3.1.

Letting $l_1(k)$ denote the inverse z-transform of $\Gamma_1(z, a)$, $l_2(k)$ the inverse z-transform of $\Gamma_2(z, a)$ and so on to $l_N(k)$ the inverse z-transform of $\Gamma_N(z, a)$. This set of discrete-time Laguerre functions are expressed in a vector form as

$$L(k) = [l_1(k) \ l_2(k) \ \dots \ l_N(k)]^T.$$

Taking advantage of the network realization (3.4), the set of discrete-time Laguerre functions satisfies the following difference equation,

$$L(k+1) = A_l L(k), \quad (3.5)$$

where matrix A_l is $(N \times N)$ and is a function of parameters a and $\beta = (1 - a^2)$, and the initial condition is given by

$$L(0)^T = \sqrt{\beta} [1 \ -a \ a^2 \ -a^3 \ \dots \ (-1)^{N-1} a^{N-1}].$$

For example, in the case where $N = 5$,

$$A_l = \begin{bmatrix} a & 0 & 0 & 0 & 0 \\ \beta & a & 0 & 0 & 0 \\ -a\beta & \beta & a & 0 & 0 \\ a^2\beta & -a\beta & \beta & a & 0 \\ -a^3\beta & a^2\beta & -a\beta & \beta & a \end{bmatrix}; \quad L(0) = \sqrt{\beta} \begin{bmatrix} 1 \\ -a \\ a^2 \\ -a^3 \\ a^4 \end{bmatrix}.$$

The orthonormality expressed in (3.2) and (3.3) also exists in the time domain, namely

$$\sum_{k=0}^{\infty} l_i(k)l_j(k) = 0 \quad \text{for } i \neq j \quad (3.6)$$

$$\sum_{k=0}^{\infty} l_i(k)l_j(k) = 1 \quad \text{for } i = j. \quad (3.7)$$

The orthonormality will be used in the design of discrete-time model predictive control.

The Special Case when $a = 0$

When $a = 0$, the A_l matrix in (3.5) becomes

$$A_l = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 1 & 0 \end{bmatrix}, \quad (3.8)$$

and the initial condition vector becomes

$$L(0)^T = [1 \ 0 \ 0 \ 0 \ \dots \ 0].$$

Then, $l_1(k) = \delta(k)$, $l_2(k) = \delta(k-1)$, $l_3(k) = \delta(k-2)$, ..., $l_N(k) = \delta(k-N+1)$, where $\delta(i) = 1$ when $i = 0$; and $\delta(i) = 0$ when $i \neq 0$. It is seen that the Laguerre functions become a set of pulses when $a = 0$. This is important because the previous work in the design of predictive control essentially uses this type of description for the incremental control trajectory, thus the MPC design using Laguerre functions with $a = 0$, becomes equivalent to the traditional approach as we discussed in Chapters 1 and 2.

Example 3.1. Suppose that the pole takes the two different locations $a = 0.5$ and $a = 0.9$, respectively, generate the first three Laguerre functions for each case. Also investigate the orthonormal property of both sets of Laguerre functions in terms of finite sums.

Solution. For a given a , the difference equation for the first three Laguerre functions is

$$\begin{bmatrix} l_1(k+1) \\ l_2(k+1) \\ l_3(k+1) \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ 1-a^2 & a & 0 \\ -a(1-a^2) & 1-a^2 & a \end{bmatrix} \begin{bmatrix} l_1(k) \\ l_2(k) \\ l_3(k) \end{bmatrix}. \quad (3.9)$$

With the initial condition $l_1(0) = \sqrt{1-a^2}$, $l_2(0) = -a\sqrt{(1-a^2)}$ and $l_3(0) = a^2\sqrt{(1-a^2)}$, the Laguerre functions are calculated iteratively. Figure 3.2 shows the Laguerre functions for $a = 0.5$ and $a = 0.9$ respectively. It is seen that with $a = 0.5$, the Laguerre functions decay to zero in less than 15 samples. By contrast, with $a = 0.9$, the Laguerre functions decay to zero at a much slower speed (approximately 50 samples are required). Also, the initial values for the Laguerre functions with the smaller a value are larger than the corresponding functions with a larger a , particularly with the first function in each set. To investigate the orthonormal property of the Laguerre functions, we calculate the finite sums, for $a = 0.5$ (S_1) and for $a = 0.9$ (S_2)

$$S_1 = \sum_{k=0}^{19} L(k)L(k)^T; \quad S_2 = \sum_{k=0}^{50} L(k)L(k)^T$$

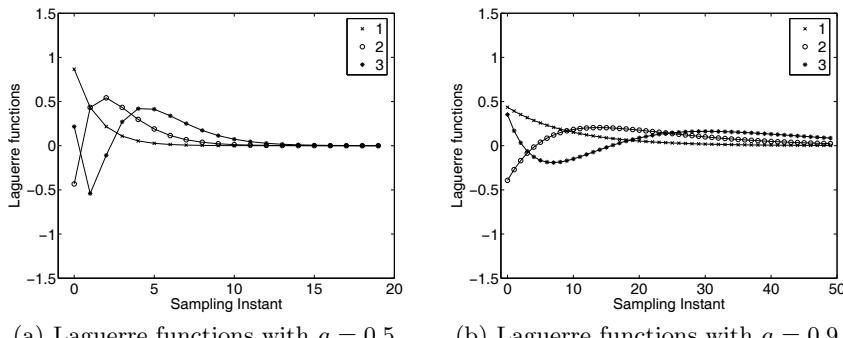
The matrices S_1 and S_2 are

$$S_1 = \begin{bmatrix} 1.0000 & -0.0000 & -0.0000 \\ -0.0000 & 1.0000 & -0.0000 \\ -0.0000 & -0.0000 & 1.0000 \end{bmatrix}; \quad S_2 = \begin{bmatrix} 1.0000 & -0.0003 & -0.0012 \\ -0.0003 & 0.9970 & -0.0129 \\ -0.0012 & -0.0129 & 0.9430 \end{bmatrix}.$$

It is seen that S_1 is an identity matrix (to 4-digit accuracy), however, S_2 is not an identity matrix. When we increase the number of samples from 50 to 90, and $a = 0.9$, we obtain that

$$S_2 = \sum_{k=0}^{90} L(k)L(k)^T = \begin{bmatrix} 1.0000 & -0.0000 & -0.0000 \\ -0.0000 & 1.0000 & -0.0000 \\ -0.0000 & -0.0000 & 0.9998 \end{bmatrix},$$

which is much closer to an identity matrix. This illustrates that the number of samples required to achieve the orthonormality is dependent on the choice of the scaling factor a and the number of terms N .



(a) Laguerre functions with $a = 0.5$

(b) Laguerre functions with $a = 0.9$

Fig. 3.2. Laguerre functions. Key: line (1) $l_1(\cdot)$; line (2) $l_2(\cdot)$; and line (3) $l_3(\cdot)$

3.2.2 Use of Laguerre Networks in System Description

The application of Laguerre networks so far is mainly in the area of system identification, where the discrete-time impulse response of a dynamic system is represented by a Laguerre model (see Wahlberg, 1991). In this regard, suppose that the impulse response of a stable system is $H(k)$, then with a given number of terms N , $H(k)$ is written as

$$H(k) = c_1 l_1(k) + c_2 l_2(k) + \dots + c_N l_N(k), \quad (3.10)$$

where c_1, c_2, \dots, c_N are the coefficients to be determined from the system data. The discrete-time Laguerre functions are orthonormal functions, and with these orthonormal properties, the coefficients of the Laguerre network are defined by the following relation, where $i = 1, 2, \dots, N$

$$c_i = \sum_{k=0}^{\infty} H(k) l_i(k).$$

In fact, because of the orthonormal properties, the coefficients also minimize the sum of the squared error function

$$J_{SE} = \sum_{k=1}^{\infty} \left(H(k) - \sum_{i=1}^N c_i l_i(k) \right)^2.$$

In a similar spirit to the approximation in the continuous-time domain (see Chapter 5), the approximation to $H(k)$ improves as the number of terms N increases, independent of the choice of the parameter $0 \leq a < 1$.

It is the description of the discrete-time impulse response that leads to the design of predictive control using Laguerre functions. The parameters in the description are the Laguerre pole location a and the number of terms N . In order to understand how to use a Laguerre network to describe an impulse response, the following MATLAB tutorial is presented so that we can experience the modelling process. We can also use this tutorial to obtain a Laguerre dynamic model for the design of discrete-time MPC.

3.2.3 MATLAB Tutorial: Use of Laguerre Functions in System Modelling

Tutorial 3.1. Suppose that a discrete-time system has the z -transfer function

$$G(z) = \frac{z - 0.1}{(z - 0.8)(z - 0.9)}.$$

Use a third order Laguerre network ($N = 3$) to approximate the impulse response of this discrete-time system. Observe how the choice of parameters N and a affects the accuracy of the approximation.

Step by Step

1. Create a new file called *lagd.m*.
2. The following program generates the initial condition of the Laguerre function $L(0)$ and the state-space system matrix A_l . Because A_l is a Toeplitz matrix, we generate its first column, and shift the first column to obtain the rest of the columns.
3. Enter the following program into the file:

```
function [A,L0]=lagd(a,N)
v(1,1)=a;
L0(1,1)=1;
for k=2:N
    v(k,1)=(-a).^(k-2)*(1-a*a);
    L0(k,1)=(-a).^(k-1);
end
L0=sqrt((1-a*a))*L0;
A(:,1)=v;
for i=2:N
    A(:,i)=[zeros(i-1,1);v(1:N-i+1,1)];
end
```

4. Create a new file called *lagdmodel.m*.
5. The discrete-time impulse response of the system is generated for the numerical experiment. MATLAB function *dimpulse.m* is used to obtain the impulse response data from a transfer function model.
6. Enter the following program into the file:

```
clear
numd=[1 -0.1];
dend=conv([1 -0.8],[1 -0.9]);
N_sim=60;
k=0:(N_sim-1);
H=dimpulse(numd,dend,k);
```

7. The discrete-time Laguerre functions are generated for a given a and N . Continue entering the following program into the file:

```
a=0.8;
N=3;
[A1,L0]=lagd(a,N);
L(:,1)=L0;
for kk=2:N_sim;
    L(:,kk)=A1*L(:,kk-1);
end
```

8. The coefficients of the Laguerre model are computed. Continue entering the following program into the file:

```

c1=L(1,:)*H;
c2=L(2,:)*H;
c3=L(3,:)*H;
H_model=c1*L(1,:)+c2*L(2,:)+c3*L(3,:);
figure
plot(k,H)
hold on
plot(k,H_model,'LineWidth',2,'Color',[.8 0 0])
set(gca,'FontSize',20,'FontName','helvetica');
legend('data','model')
xlabel('Sampling Instant')
ylabel('Impulse Response')

```

9. Run the program *lagdmodel.m* to produce the response of the Laguerre model for comparison with the actual data.
10. You may check the orthonormality of the Laguerre functions by typing in the work space

```

L(:,1)*L(:,1)'
L(:,1)*L(:,2)'

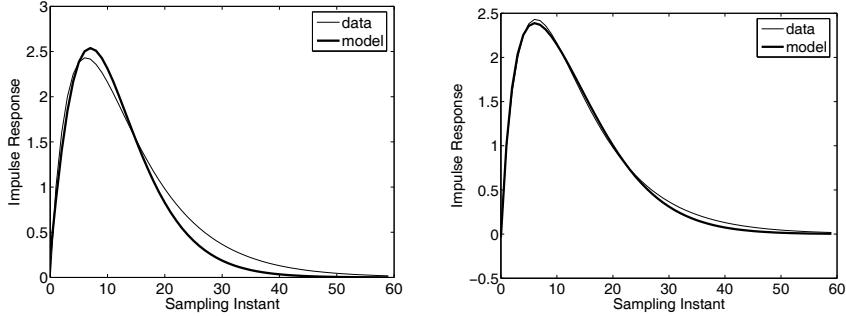
```

11. Try increasing the parameter N to 4 and see the improvement of the modelling results.
12. Try to determine the effect of the pole location a on the modelling results. For instance, try $a = 0$.

Figure 3.3 shows the modelling results for $N = 3$ and $N = 4$ with $a = 0.8$. When N increases to 4, the modelling result is improved. One observation is that when $N = 4$, there are only 4 parameters to be used in capturing the dynamics of the impulse response. In contrast, in the case of $a = 0$, *i.e.*, the pulse operator, 60 parameters are required to capture the dynamics of the system. Therefore, it is understandable that when an appropriate a is selected, the Laguerre network is much more efficient than the pulse operator model (*i.e.*, $a = 0$).

3.3 Use of Laguerre Functions in DMPC Design

The previous chapters saw the design of DMPC using the pulse operator (*i.e.*, ΔU), which corresponds to the case where the parameter $a = 0$ in the Laguerre polynomial. As a consequence, in the case of rapid sampling, complicated process dynamics and /or high demands on closed-loop performance, satisfactory approximation of the control signal Δu may require a very large number of parameters, leading to poorly numerically conditioned solutions and heavy computational load when implemented on-line. Instead, a more appropriate approach is to use Laguerre networks in the design of model predictive control. For example, supposing that the underlying optimal control



(a) Modelling results with $N = 3$ and $a = 0.8$ (b) Modelling results with $N = 4$ and $a = 0.8$

Fig. 3.3. Comparison between actual data and Laguerre model response

trajectory was the impulse response of the second-order system in Figure 3.3, then when $a = 0$ which was the case of the pulse operator, there are 60 parameters required to capture the response. However, with the Laguerre polynomial with $a = 0.8$, there were only 4 parameters required to perform the same task.

3.3.1 Design Framework

At time k_i , the control trajectory $\Delta u(k_i), \Delta u(k_i + 1), \Delta u(k_i + 2), \dots, \Delta u(k_i + k), \dots$, is regarded as the impulse response of a stable dynamic system. Thus, a set of Laguerre functions, $l_1(k), l_2(k), \dots, l_N(k)$ are used to capture this dynamic response with a set of Laguerre coefficients that will be determined from the design process. More precisely, at an arbitrary future sample instant k ,

$$\Delta u(k_i + k) = \sum_{j=1}^N c_j(k_i) l_j(k), \quad (3.11)$$

with k_i being the initial time of the moving horizon window and k being the future sampling instant; N is the number of terms used in the expansion and $c_j, j = 1, 2, \dots, N$, are the coefficients, and they are functions of the initial time of the moving horizon window, k_i . Within this design framework, the control horizon N_c from the previous approach has vanished. Instead, the number of terms N is used to describe the complexity of the trajectory in conjunction with the parameter a . For instance, a larger value of a can be selected to achieve a long control horizon with a smaller number of parameters N required in the optimization procedure. We note that when $a = 0$, $N = N_c$, and we recover the traditional approach of previous chapters.

Equation (3.11) may also be written in a vector form:

$$\Delta u(k_i + k) = L(k)^T \eta, \quad (3.12)$$

where the parameter vector η comprises N Laguerre coefficients:

$$\eta = [c_1 \ c_2 \ \dots \ c_N]^T,$$

and $L(k)^T$ is the transposed Laguerre function vector as defined in the difference equation (3.5). When using Laguerre functions, given the state-space model (A, B, C) with $\Delta u(\cdot)$ as the input signal with the initial state variable information $x(k_i)$, the prediction of the future state variable, $x(k_i + m | k_i)$ at sampling instant m , becomes

$$x(k_i + m | k_i) = A^m x(k_i) + \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \eta, \quad (3.13)$$

where we replaced the function $\Delta u(k_i + i)$ by $L(i)^T \eta$. Similarly, upon obtaining the prediction of state variables, the prediction for the plant output at future sample m is

$$y(k_i + m | k_i) = C A^m x(k_i) + \sum_{i=0}^{m-1} C A^{m-i-1} B L(i)^T \eta. \quad (3.14)$$

With this formulation, both predictions of state variable and output variable are expressed in terms of the coefficient vector η of the Laguerre network, instead of ΔU as in the more traditional approach. Thus, the coefficient vector η will be optimized and computed in the design.

3.3.2 Cost Functions

Although the model predictive control algorithms in Chapters 1 and 2, presented in matrix vector forms, are easy to understand and the implementation is simple, it requires a large computer memory to form and store the system data matrices. Here, with the Laguerre functions, an alternative formulation of the cost function is obtained, and this may be easier from the programming point of view, particularly for systems with a large number of inputs and outputs. In the first instant, let us keep things simple by assuming a SISO system. Recall from Chapter 1 that at time k_i the set-point signal, R_s is a constant vector within the prediction horizon, and the original cost function is

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T \bar{R} \Delta U,$$

where

$$Y = [y(k_i + 1 | k_i) \ y(k_i + 2 | k_i) \ y(k_i + 3 | k_i) \ \dots \ y(k_i + N_p | k_i)]^T,$$

$$\Delta U = [\Delta u(k_i) \ \Delta u(k_i + 1) \ \Delta u(k_i + 2) \ \dots \ \Delta u(k_i + N_c - 1)]^T,$$

$$R_s^T = \overbrace{[1 \ 1 \ \dots \ 1]}^{N_p} r(k_i),$$

and \bar{R} is a diagonal matrix with identical element r_w . Noting that Y and ΔU are in a vector form, this cost function is equivalent to

$$J = \sum_{m=1}^{N_p} (r(k_i) - y(k_i + m | k_i))^T (r(k_i) - y(k_i + m | k_i)) + \eta^T R_L \eta, \quad (3.15)$$

where R_L is a diagonal matrix ($N \times N$) with an $r_w \geq 0$ on its diagonal and $r(k_i)$ is the set-point signal for the output y at time k_i . In (3.15), we have used the orthonormal properties of the Laguerre functions in the way that

$$\Delta U^T \bar{R} \Delta U = \sum_{m=0}^{N_p} \Delta u(k_i + m)^T r_w \Delta u(k_i + m),$$

with

$$\Delta u(k_i + m) = [l_1(m) \ l_2(m) \ \dots \ l_N(m)] \eta$$

and the orthonormal property of the Laguerre functions (see (3.6) and (3.7)) with a sufficiently large prediction horizon N_p so that

$$\begin{aligned} \sum_{m=0}^{N_p} l_i(m) l_j(m) &= 0 \text{ for } i \neq j \\ \sum_{m=0}^{N_p} l_i(m) l_j(m) &= 1 \text{ for } i = j. \end{aligned}$$

Here, the large value of N_p is related to the choice of both the scaling factor a and N . We have investigated this issue in Example 3.1.

The cost function (3.15) is based on the minimization of the error between the set-point signal and the output signal. The reasons for this choice include simplicity, practicality of the cost and its relevance to applications, and its similarity to the classical predictive control systems. Here, the cost function is re-formulated with a link to discrete-time linear quadratic regulators (DLQR) (see (3.36) to (3.38)), where the objective is to find the coefficient vector η to minimize the cost function:

$$J = \sum_{m=1}^{N_p} x(k_i + m | k_i)^T Q x(k_i + m | k_i) + \eta^T R_L \eta, \quad (3.16)$$

with the weighting matrices $Q \geq 0$ and $R_L > 0$. In particular, Q has the dimension equal to the number of state variables and R_L has the dimension equal to dimension η . The reason for this re-formulation is to connect the discrete-time MPC with the DLQR system so that the numerous classical results in DLQR will lend themselves to the analysis, tuning and design of discrete-time MPC. To this end, let us compare the cost function (3.16) with the cost function (3.15).

Regulator Design where the Set-point $r(k) = 0$

When the set-point signal $r(k) = 0$, if Q is chosen to be $C^T C$, then the cost (3.16) is identical to (3.15). Note that the set-point signal $r(k)$ represents an incremental change of the plant operating condition, where $r(k) = 0$ means that the plant output maintains its steady-state operation. Thus, traditionally, the cost (3.16) is often used in regulator design where the purpose of the control is to maintain closed-loop stability and reject disturbances occurring in the plant.

Inclusion of Set-point Signal $r(k) \neq 0$

In order to include the set-point signal in the cost function (3.16), we need to re-define the state variable $x(k_i + m | k_i)$. Recall that in the original augmented model, the output matrix C is defined as

$$C = [0 \ 0 \ \dots \ 0 \ 1],$$

and the state variable vector $x(k) = [\Delta x_m(k)^T \ y(k)]^T$. We choose a vector $x_r(k_i)$, where $x_r(k_i)$ is defined by

$$x_r(k_i) = [0 \ 0 \ \dots \ 0 \ r(k_i)]^T,$$

which has the same dimension as the augmented state variable $x(.)$, with the number of zeros identical to the dimension of $\Delta x_m(k)$. This will lead to $r(k_i) = C x_r(k_i)$. Also, note that in the prediction and optimization, the set-point signal $r(k_i)$ remains unchanged within the optimization window. Thus, with the inclusion of a set-point signal $r(k_i)$, within the optimization window, if the state variable $x(k_i + m | k_i)$ is chosen to be

$$x(k_i + m | k_i) = [\Delta x_m(k_i + m | k_i)^T \ y(k_i + m | k_i) - r(k_i)]^T,$$

then the cost function for minimization of output errors (see (3.15)) is identical to the cost function:

$$J = \sum_{m=1}^{N_p} x(k_i + m | k_i)^T Q x(k_i + m | k_i) + \eta^T R_L \eta, \quad (3.17)$$

with $Q = C^T C$, which is (3.16). It is emphasized that this inclusion of the set-point signal in the cost function does not alter the model used for prediction.

3.3.3 Minimization of the Objective Function

Recall from (3.13) that

$$\begin{aligned} x(k_i + m \mid k_i) &= A^m x(k_i) + \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \eta \\ &= A^m x(k_i) + \phi(m)^T \eta, \end{aligned} \quad (3.18)$$

where the matrix $\phi(m)^T = \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T$.

Note that the number of rows in $\phi(m)$ is identical to the number of rows in η . By substituting (3.18) into the cost function (3.16), we obtain

$$\begin{aligned} J &= \eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \eta \\ &\quad + 2\eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i) + \sum_{m=1}^{N_p} x(k_i)^T (A^T)^m Q A^m x(k_i). \end{aligned} \quad (3.19)$$

To find the minimum of (3.19), without constraints, we use the partial derivative of the cost function:

$$\frac{\partial J}{\partial \eta} = 2 \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \eta + 2 \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i). \quad (3.20)$$

Assuming that $\left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right)^{-1}$ exists, when $\frac{\partial J}{\partial \eta} = 0$, the optimal solution of the parameter vector η is

$$\eta = - \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right)^{-1} \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i). \quad (3.21)$$

For simplicity of the expression, we define

$$\Omega = \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \quad (3.22)$$

$$\Psi = \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right), \quad (3.23)$$

leading to

$$\eta = -\Omega^{-1} \Psi x(k_i). \quad (3.24)$$

The Minimum of the Cost

The minimum value of the cost, J_{min} , is of interest to us. To find the minimum, we complete the square

$$\begin{aligned}
J &= \eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \eta \\
&\quad + 2\eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i) + \sum_{m=1}^{N_p} x(k_i)^T (A^T)^m Q A^m x(k_i) \quad (3.25) \\
&= (\eta + \Omega^{-1} \Psi x(k_i))^T \Omega (\eta + \Omega^{-1} \Psi x(k_i)) \\
&\quad - x(k_i)^T \Psi^T \Omega^{-1} \Psi x(k_i) + \sum_{m=1}^{N_p} x(k_i)^T (A^T)^m Q A^m x(k_i). \quad (3.26)
\end{aligned}$$

Therefore, with the optimal solution (3.21), the minimum of the cost function is

$$J_{min} = x(k_i)^T \left(\sum_{m=1}^{N_p} (A^T)^m Q A^m - \Psi^T \Omega^{-1} \Psi \right) x(k_i) \quad (3.27)$$

$$= x(k_i)^T P_{dmpc} x(k_i), \quad (3.28)$$

where the matrix $P_{dmpc} = \sum_{m=1}^{N_p} (A^T)^m Q A^m - \Psi^T \Omega^{-1} \Psi$.

3.3.4 Convolution Sum

To compute the prediction, the convolution sum

$$S_c(m) = \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T$$

needs to be computed. To this end, note that

$$\begin{aligned}
S_c(1) &= B L(0)^T \\
S_c(2) &= A B L(0)^T + B L(1)^T = A B L(0)^T + B L(0)^T A_l^T = A S_c(1) + S_c(1) A_l^T \\
S_c(3) &= A^2 B L(0)^T + A B L(1)^T + B L(2)^T \\
&= A S_c(2) + S_c(1) (A_l^2)^T, \quad (3.29)
\end{aligned}$$

where we have used the relation $L(k+1) = A_l L(k)$ which is the difference equation for generating the set of Laguerre functions in (3.5). Continuing the recursion in (3.29) reveals that

$$S_c(m) = A S_c(m-1) + S_c(1) (A_l^{m-1})^T, \quad (3.30)$$

where $S_c(1) = B L(0)^T$ and $m = 2, 3, 4, \dots, N_p$. Note that in the computation of $S_c(\cdot)$, the Laguerre functions are defined through its state-space formulation. For a given a and N , A_l and $L(0)$ are defined according to (3.5).

3.3.5 Receding Horizon Control

Upon obtaining the optimal parameter vector η , the receding horizon control law is realized as

$$\Delta u(k_i) = L(0)^T \eta, \quad (3.31)$$

where for a given N and a ,

$$L(0)^T = \sqrt{(1-a^2)} [1 \ -a \ a^2 \ -a^3 \ \cdots \ (-1)^{N-1} a^{N-1}].$$

The control $\Delta u(k)$ can be written in the form of linear state feedback control by replacing k_i with k (η is a function of state variable $x(k_i)$). Namely,

$$\Delta u(k) = -K_{mpc}x(k), \quad (3.32)$$

and the state feedback gain matrix

$$K_{mpc} = L(0)^T \left((\sum_{m=1}^{N_p} \phi(m)Q\phi(m)^T + R_L)^{-1} \sum_{m=1}^{N_p} \phi(m)QA^m \right),$$

which is simplified in notation as

$$K_{mpc} = L(0)^T \Omega^{-1} \Psi.$$

Note that in the case when $a = 0$ (see (3.8)), the solution becomes identical to the case where we solve ΔU instead of the Laguerre coefficient vector η .

Since the prediction of future states is based on the current information on $x(k_i)$, the set-point information is contained in $x(k_i)$. More specifically

$$\begin{aligned} x(k_i) &= [\Delta x_m(k_i)^T \ e(k_i)]^T \\ e(k_i) &= y(k_i) - r(k_i). \end{aligned}$$

When an observer is used in the design, the information of the observed states replaces the actual state variables.

3.3.6 The Optimal Trajectory of Incremental Control

In order to understand the behaviour of the DMPC, we present the following example, which examines the trajectory of the optimal trajectory of $\Delta u(k_i + m)$, for $m = 0, 1, 2, \dots$

Example 3.2. Suppose that a first-order system is described by the state equation

$$\begin{aligned} x_m(k+1) &= 0.8x_m(k) + 0.6u(k) \\ y(k) &= x_m(k). \end{aligned} \quad (3.33)$$

Given at time $k_i = 10$, the state vector $x(k_i) = [0.1 \ 0.2]^T$ and a prediction horizon $N_p = 16$, find the optimal trajectory of $\Delta u(k_i + m)$ for $m = 0, 1, 2, \dots, N_p$. The design parameters are $Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$, $R = 1$, and $a = 0.6$. Examine solutions where N increases from 1 to 4.

Table 3.1. Parameters in η

N	c_1	c_2	c_3	c_4
1	-0.1149			
2	-0.1734	0.0804		
3	-0.1844	0.1120	-0.0293	
4	-0.1842	0.1110	-0.0274	-0.0022

Solution. The augmented state-space equation is

$$\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} 0.8 & 0 \\ 0.8 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} 0.6 \\ 0.6 \end{bmatrix} \Delta u(k). \quad (3.34)$$

With the initial state variable at $k_i = 10$ given as $x(k_i) = [0.1 \ 0.2]^T$, the optimal solution of the parameter vector η is

$$\eta = - \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right)^{-1} \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i).$$

The coefficient vector η is presented in the Table 3.1 for $N = 1, 2, 3, 4$. With the coefficients from Table 3.1, the optimal control trajectory $\Delta u(k_i + m)$ is constructed as, $m = 0, 1, 2, \dots, 16$,

$$\Delta u(k_i + m) = \sum_{j=1}^N c_j l_j(m).$$

Figure 3.4a shows $\Delta u(k_i + m)$ for $k_i = 10$, and $m = 0, 1, 2, \dots$. It is seen from this figure that the incremental control signal is almost identical for $N = 3$ and $N = 4$. This is confirmed from the output responses for the two cases, as shown in Figure 3.4b. However, when $N = 1$, the control trajectory is very different from the trajectory of $N = 3$ or 4 . The output response from $N = 1$ is much slower than the other three cases, because the control signal is restricted to the first-order response.

Convergence of the Incremental Control Trajectory

Assuming a large prediction horizon N_p , we understand from Section 3.2 that for a given pole a , as N increases, the trajectory $\Delta u(\cdot)$ captured by the Laguerre functions will converge to the function that is being modelled. What is this function and how can we define it? This underlying function is the incremental control trajectory $\Delta u(m)$ with a given initial condition $x(k_i)$, defined by the optimal minimizing solution of

$$J = \sum_{m=1}^{\infty} x(k_i + m | k_i)^T Q x(k_i + m | k_i) + \sum_{m=0}^{\infty} \Delta u(k_i + m)^T R \Delta u(k_i + m). \quad (3.35)$$

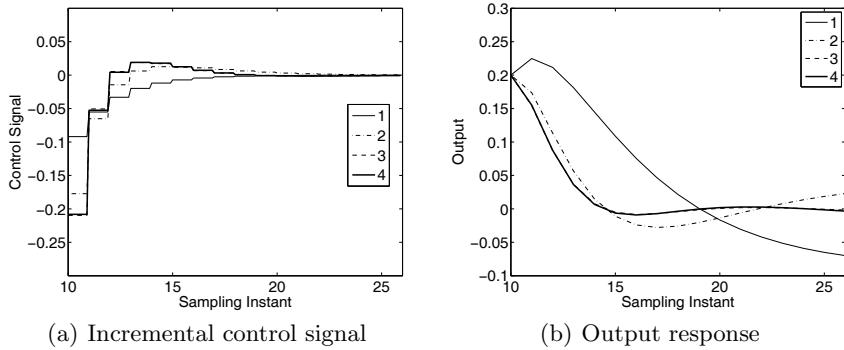


Fig. 3.4. Comparison of responses within one optimization window. Key: line (1) $N = 1$; line (2) $N = 2$; line (3) $N = 3$; line (4) $N = 4$.

The optimal controller that minimizes the cost function is also called a discrete-time linear quadratic regulator (DLQR) (Kailath, 1980, Bay, 1999, Grimble and Johnson, 1988a). The solution for this discrete-time regulator is found through the following steps. Firstly the algebraic Riccati equation (3.36) is solved for the Riccati matrix P_∞ ; secondly, the matrix P_∞ is used to construct the state feedback gain matrix K using (3.37); thirdly the feedback control signal is defined using K and the state variable $x(k)$ using (3.38).

$$A^T (P_\infty - P_\infty B (R + B^T P_\infty B)^{-1} B^T P_\infty) A + Q - P_\infty = 0 \quad (3.36)$$

$$K = (R + B^T P_\infty B)^{-1} B^T P_\infty A \quad (3.37)$$

$$\Delta u(k) = -K x(k). \quad (3.38)$$

With the given initial condition $x(k_i)$, this optimal control $\Delta u(k)$, $k = 0, 1, 2, \dots$, is the control trajectory that the Laguerre functions are to capture when N is large. This control trajectory is unique for a specific choice of Q and R matrices.

Let us illustrate these points using the following example.

Example 3.3. Illustrate that as N increases, for a given Laguerre pole a , the control trajectory converges to the optimal solution generated by using a linear quadratic regulator. Use the same system and design parameters as in Example 3.2, where the augmented system model is described by the matrices,

$$A = \begin{bmatrix} 0.8 & 0 \\ 0.8 & 1 \end{bmatrix}; \quad B = \begin{bmatrix} 0.6 \\ 0.6 \end{bmatrix}; \quad C = [0 \ 1].$$

Table 3.2. Sum of squared errors between the control trajectories

N	1	2	3	5
E	0.0179	0.0018	1.384×10^{-4}	7.3913×10^{-6}

Solution. We use the MATLAB program called ‘dlqr’ to find the state-feedback control gain matrix K as

$$K = [0.8350 \ 0.6113].$$

for which we use the MATLAB Script

```
[K,P,Z]=dlqr(A,B,Q,R);
```

where $Q = C^T C$ and $R = 1$. Then the optimal control trajectory from the linear quadratic regulator is described by

$$\Delta u(k_i + m) = -K(A - BK)^m x(k_i),$$

where $m = 0, 1, 2, \dots$

We use $a = 0.6$, and compute the set of the control trajectories with $N = 1, 2, 3$ and 5. The sum of squared errors between the DLQR and DMPC trajectories is calculated as $E = \|\Delta u_{lqr} - \Delta u_{mpc}\|_2^2$, where Table 3.2 shows how the error decreases as N increases. Furthermore, Figure 3.5a compares the control trajectory obtained from DLQR with the lower-order Laguerre description ($N = 3, a = 0.6$), where we can see there is a difference between the two trajectories. However, by increasing N to 5, the two control trajectories become almost identical as shown in Figure 3.5b.

To illustrate that the performance is determined by Q and R matrices, we examine the situation when R varies, while the matrix Q is selected as $C^T C$.

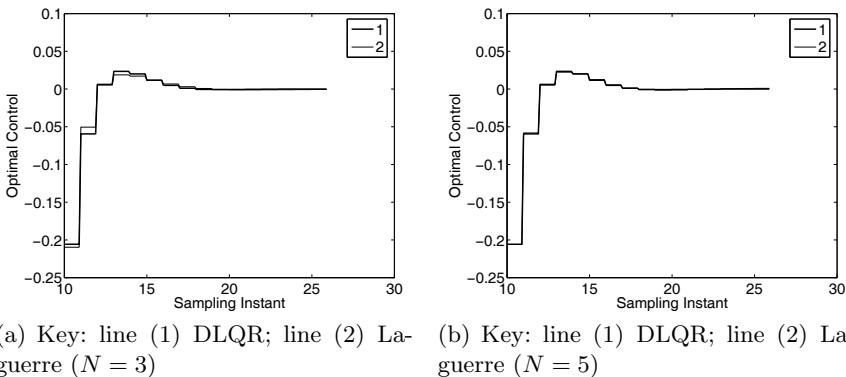


Fig. 3.5. Comparison of Δu within one optimization window using DLQR and using Laguerre functions

Example 3.4. Continue the study from Example 3.3 with the identical system and design parameters, except that the weight on the control is reduced from $R = 1$ to $R = 0.1$. Examine the incremental control trajectory $\Delta u(k_i + m)$ for this performance specification, and compare the trajectory with the one from DLQR solution.

Solution. With $a = 0.6$, $N = 5$, $R = 0.1$, $N_p = 16$,

$$x(k_i) = [0.1 \ 0.2]^T; \quad Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix},$$

the Laguerre coefficient vector η is calculated as

$$\eta = -\left(\sum_{m=1}^{N_p} \phi(m)Q\phi(m)^T + R_L\right)^{-1} \left(\sum_{m=1}^{N_p} \phi(m)QA^m\right)x(k_i),$$

which gives the value of η as

$$\eta = [-0.2253 \ 0.2189 \ -0.1486 \ 0.0777 \ -0.0354]^T.$$

Using η , the incremental control trajectory is constructed as

$$\Delta u(k_i + m) = L(m)^T \eta,$$

where $L(m)$ is the Laguerre function vector.

In comparison with the optimal control trajectory from DLQR, the optimal control gain vector is found using the performance matrices Q and R , with the value

$$K = [1.1485 \ 1.1775],$$

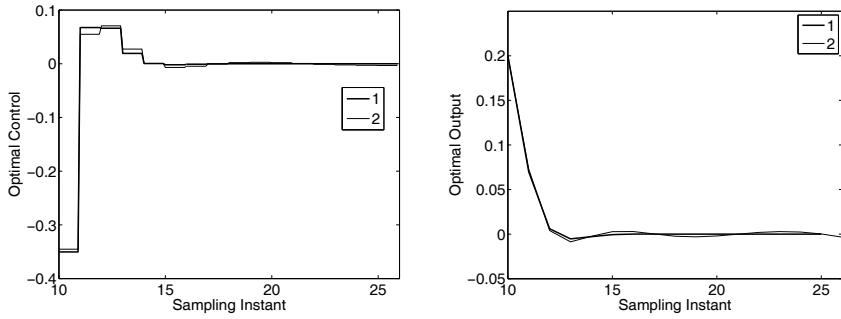
and the control trajectory is generated using

$$\Delta u(k_i + m) = -K(A - BK)^m x(k_i).$$

With a smaller weight $R = 0.1$, the feedback control gain K has increased from that in Example 3.3. The DLQR control trajectory has also changed from that in Figure 3.5, where faster response speed is observed. Figure 3.6 shows the comparison results between the Laguerre based DMPC and the DLQR. The sum of squared errors between the control trajectories is calculated as $E = \|\Delta u_{lqr} - \Delta u_{mpc}\|_2^2 = 3.44 \times 10^{-4}$, which is larger than the previous case, indicating that the scaling factor $a = 0.6$ is not as suitable for this case because the control trajectory has a faster decay rate. However, the difference is very small.

Closed-loop Poles of a DLQR System

In the DLQR design, when we choose $Q = C^T C$, the closed-loop poles are uniquely determined by the weighting parameter $R = r_w > 0$ in the single-input and single-output case (see Kailath, 1980) with their values given by the inside-the-unit-circle zeros of the equation,

(a) Key: line (1) DLQR ; line (2) La-
guerre ($N = 5$)(b) Key: line (1) DLQR; line (2) La-
guerre**Fig. 3.6.** Comparison of Δu and output response within one optimization window using DLQR and using Laguerre functions

$$1 + \frac{1}{r_w} \frac{G_m(z)G_m(z^{-1})}{(z-1)(z^{-1}-1)} = 0. \quad (3.39)$$

In the multi-input and multi-output case, when the weighting matrices $Q = C^T C$, and $R = r_w I$, the closed-loop poles are the inside-the-unit-circle zeros of the equation,

$$\det(I + \frac{1}{r_w} \frac{G_m(z)G_m(z^{-1})^T}{(z-1)(z^{-1}-1)}) = 0,$$

where $G_m(z) = C_m(zI - A_m)^{-1}B_m$ is the z-transfer function for the plant.

We verify this property by re-examining solutions from the previous examples 3.2, 3.3, 3.4. The plant z-transfer function used in these examples is

$$G_m(z) = \frac{0.6}{z - 0.8}.$$

The equation corresponding to (3.39) for $R = 1$ is

$$1 + \frac{0.36}{(z - 0.8)(z^{-1} - 0.8)(z - 1)(z^{-1} - 1)} = 0. \quad (3.40)$$

The zeros are $1.5589 \pm j0.9563$, $0.4661 \pm j0.2859$ from which the closed-loop poles of the DLQR system are found to be $0.4661 \pm j0.2859$.

When we reduced R from 1 to 0.1, the equation corresponding to (3.39) is

$$1 + \frac{3.6}{(z - 0.8)(z^{-1} - 0.8)(z - 1)(z^{-1} - 1)} = 0. \quad (3.41)$$

The zeros are $1.8228 \pm j2.3858$, $0.2022 \pm j0.2647$ from which the closed-loop poles of the DLQR system are found to be $0.2022 \pm j0.2647$. Indeed, these pole

locations are confirmed with the computation of the actual closed-loop poles through the system matrix $A - BK_{lqr}$. In addition, we can see that when the weight parameter R is reduced, the closed-loop poles move towards the origin of the complex plane, hence the closed-loop response speed is faster, as we have observed from the responses in the examples.

Use of Laguerre Parameters as Tuning Parameters

The choice of a scaling factor a and a smaller N for the Laguerre functions affects the closed-loop performance of the DMPC system, although for a large N the performance converges to the underlying optimal DLQR system. However, when N is small, the potential here is to use a as the performance fine-tuning ‘knob’. To illustrate these points, we consider the following example.

Example 3.5. Suppose that a continuous-time system is described by the transfer function

$$G(s) = \frac{s - 3}{s^2 + 2\xi\omega s + \omega^2},$$

where $\xi = 0.3$ and $\omega = 3$. Suppose that the sampling interval $\Delta t = 0.1$, and the weight matrices are $Q = C^T C$ with $C = [0 \ 0 \ 1]$ and $R = 0.3$. The prediction horizon is chosen to be $N_p = 36$. Examine the effect of parameter a and N on the closed-loop performance. Assume at time $k_i = 10$, $x(k_i) = [0.1 \ 0.2 \ 0.3]^T$.

Solution. With $\Delta t = 0.1$, the augmented discrete-time state-space model is

$$x(k+1) = \begin{bmatrix} 0.7956 & -0.8114 & 0 \\ 0.0902 & 0.9579 & 0 \\ 0.5252 & -3.6850 & 1.0000 \end{bmatrix} x(k) + \begin{bmatrix} 0.0902 \\ 0.0047 \\ 0.0761 \end{bmatrix} \Delta u(k).$$

With $R = 0.3$, the closed-loop poles of the DLQR system are $0.7133 \pm j0.3058$, and 0.7867 with feedback gain $K_{lqr} = [5.5331 \ 31.2042 \ -1.3752]$. Let us look at two scenarios.

Case A. We fix the parameter N with a large value ($N = 8$), and vary the pole location a by choosing $a = 0$, $a = 0.4$, and $a = 0.8$. Table 3.3 shows the comparison results for the cases where N is large while a changes. It is seen from Table 3.3 that the closed-loop predictive control system is very similar to the underlying DLQR system and the effect of a is quite small when N is large.

Case B. For the second case, we fix N with a smaller value ($N = 2$), and vary the parameter a . Table 3.4 summarizes the comparison results. It is seen that with a smaller N , the closed-loop poles and gain of the predictive control system are functions of the Laguerre pole a .

Table 3.3. Closed-loop eigenvalues and feedback gain vector when $N = 8$ and a varies from 0 to 0.8

a	closed-loop eigenvalues	feedback gain
0	$0.7082 \pm j0.3098, 0.7749$	[5.7647 33.4547 - 1.4962]
0.4	$0.7133 \pm j0.3058, 0.7868$	[5.5339 31.2102 - 1.3753]
0.8	$0.7144 \pm j0.3066, 0.7867$	[5.5118 31.1298 - 1.3740]

Table 3.4. Closed-loop eigenvalues and feedback gain vector when $N = 2$ and a varies from 0 to 0.9

a	closed-loop eigenvalues	feedback gain
0	$0.7772 \pm j0.2359, 0.5955$	[6.4138 30.8348 - 1.5625]
0.4	$0.6333 \pm j0.3069, 0.8523$	[6.2820 34.7725 - 1.2397]
0.8	$0.7980 \pm j0.1978, 0.7875$	[3.9763 12.6001 - 0.6232]
0.9	$0.8498 \pm j0.2298i, 0.8302$	[2.4923 7.4413 - 0.4697]

We observe that the parameters a and N affect the closed-loop control performance, particularly when N is small. This is particularly useful in the situation when the optimal DLQR system does not provide us with satisfactory performance, and these additional tuning parameters help us with fine tuning of the closed-loop performance.

3.4 Extension to MIMO Systems

In the formulation of MIMO predictive control system, each input signal is designated to have a Laguerre pole location independently so that its pole location a can be used to influence the decay rate of the incremental control signal. For instance, if we want a specific incremental control to decay fast, then we choose its pole location $a = 0$, otherwise, a is chosen to be greater than zero. Bearing this in mind, the description is extended to multi-input systems with full flexibility in the choice of a and N parameters. Let

$$\Delta u(k) = [\Delta u_1(k) \Delta u_2(k) \dots \Delta u_m(k)]^T,$$

and the input matrix be partitioned to

$$B = [B_1 \ B_2 \ \dots \ B_m],$$

where m is the number of inputs and B_i is the i th column of the B matrix. We express the i th control signal $\Delta u_i(k)$ by choosing a scaling factor a_i and order N_i , where a_i and N_i are selected for this particular input, such that

$$\Delta u_i(k) = L_i(k)^T \eta_i, \quad (3.42)$$

where η_i and $L_i(k)$ are the Laguerre network description of the i th control, specifically

$$L_i(k)^T = [l_1^i(k) \ l_2^i(k) \ \dots \ l_{N_i}^i(k)].$$

Based on the partition of the input matrix and given state variable information at $x(k_i)$, the prediction of the future state at time m is written as

$$\begin{aligned} x(k_i + m \mid k_i) &= A^m x(k_i) \\ &+ \sum_{j=0}^{m-1} A^{m-j-1} [B_1 L_1(j)^T \ B_2 L_2(j)^T \ \dots \ B_m L_m(j)^T] \eta \\ &= A^m x(k_i) + \phi(m)^T \eta, \end{aligned} \quad (3.43)$$

where the parameter vector η and the data matrix $\phi(m)^T$ consist of the individual coefficient vectors given by

$$\begin{aligned} \eta^T &= [\eta_1^T \ \eta_2^T \ \dots \ \eta_m^T] \\ \phi(m)^T &= \sum_{j=0}^{m-1} A^{m-j-1} [B_1 L_1(j)^T \ B_2 L_2(j)^T \ \dots \ B_m L_m(j)^T]. \end{aligned}$$

Note that the k th block matrix

$$\phi(m)_k^T = \sum_{j=0}^{m-1} A^{m-j-1} B_k L_k(j)^T$$

has the identical structure as the single-input case defined by $S_c(m)$, thus it can be computed recursively using (3.30). From here on, the convolution sum in a multi-input system is decomposed into computing the subsystems, and the computed results are put together block by block to form the multi-input structure. We emphasize that in the formulation of the multivariable problem, the scaling factors a_i and the number of terms N_i can be chosen independently for each input signal.

Similar to the SISO case, the cost function is defined as

$$\begin{aligned} J &= \eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \eta \\ &+ 2\eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i) + \sum_{m=1}^{N_p} x(k_i)^T (A^T)^m Q A^m x(k_i) \\ &= \eta^T \Omega \eta + 2\eta^T \Psi x(k_i) + \sum_{m=1}^{N_p} x(k_i)^T (A^T)^m Q A^m x(k_i), \end{aligned} \quad (3.44)$$

where the matrices Ω and Ψ are

$$\Omega = \sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L; \quad \Psi = \sum_{m=1}^{N_p} \phi(m) Q A^m.$$

Without constraints, the optimal solution of the cost function (3.44) is given by

$$\eta = -\Omega^{-1}\Psi x(k_i). \quad (3.45)$$

Upon obtaining the optimal parameter vector η , the receding horizon control law is realised as

$$\Delta u(k_i) = \begin{bmatrix} L_1(0)^T & o_2^T & \dots & o_m^T \\ o_1^T & L_2(0)^T & \dots & o_m^T \\ \vdots & \vdots & \ddots & \vdots \\ o_1^T & o_2^T & \dots & L_m(0)^T \end{bmatrix} \eta. \quad (3.46)$$

where o_k^T , $k = 1, 2, \dots, m$ represents a zero block row vector with identical dimension to $L_k(0)^T$. The control variable $\Delta u(k)$ can be written in the form of linear state feedback control by replacing k_i with k and assuming that the future reference trajectories are constant within the prediction horizon. Namely,

$$\Delta u(k) = -K_{mpc}x(k), \quad (3.47)$$

where the state feedback control gain matrix K_{mpc} is

$$K_{mpc} = \begin{bmatrix} L_1(0)^T & o_2^T & \dots & o_m^T \\ o_1^T & L_2(0)^T & \dots & o_m^T \\ \vdots & \vdots & \ddots & \vdots \\ o_1^T & o_2^T & \dots & L_m(0)^T \end{bmatrix} \Omega^{-1}\Psi.$$

With the definition of state feedback control gain matrix K_{mpc} , the closed-loop feedback control is

$$x(k+1) = (A - BK_{mpc})x(k), \quad (3.48)$$

and closed-loop stability and performance of the predictive control system can be checked by examining the location of its eigenvalues.

3.5 MATLAB Tutorial Notes

3.5.1 DMPC Computation

This is one of the most important tutorials. The function created in this exercise will be used for other applications through the remaining chapters in discrete-time systems. Also, the examples presented before can be repeated using this function.

Tutorial 3.2. Write a MATLAB function *dmpc.m* for generating the data matrices in the design of discrete-time model predictive control system where the cost function is expressed as

$$J = \eta^T E \eta + 2\eta^T H x(k_i),$$

with Laguerre functions for a multi-input system with weight matrix Q on state variables and R on the input variables.

Step by Step

1. Create a new file called *dmpc.m*.
2. The following program sets the initial conditions and dimensionality of the state, input and output variables. It also translates the weight matrix R on each input variable to the diagonal weight on the parameter vector η .
3. Enter the following program into the file:

```
function [E,H]=dmpc(A_e,B_e,a,N,Np,Q,R);
%A_e;B_e define the extended state-space model when
% integrator is used
%they can also be other forms of state-space models
% a contains the Laguerre pole locations for each input
%N the number of terms for each input
%Np prediction horizon
%Q weight on the state variables
%R weight on the input variables assumed to be diagonal.
% The cost function is J= eta ^T E eta +2 eta ^T H x(k_i)
[n,n_in]=size(B_e);
N_pa=sum(N); %the dimension of eta
E=zeros(N_pa,N_pa);
H=zeros(N_pa,n);
R_para=zeros(N_pa,N_pa);
n0=1;
ne=N(1);
for i=1:n_in-1;
R_para(n0:ne,n0:ne)= R(i,i)*eye(N(i),N(i));
n0=n0+N(i);
ne=ne+N(i+1);
end
R_para(n0:N_pa,n0:N_pa)=R(n_in,n_in)*eye(N(n_in),N(n_in));
```

4. The program below prepares the initial conditions for the convolution sum and calculates the case $i = 1$. Each input takes a position in the convolution sum.

5. Continue entering the following program into the file:

```
S_in=zeros(n,N_pa);
[A1,L0]=lagd(a(1),N(1));
S_in(:,1:N(1))=B_e(:,1)*L0';
In_s=1;
```

```

for jj=2:n_in;
[A1,LO]=lagd(a(jj),N(jj));
In_s=N(jj-1)+In_s;
In_e=In_s+N(jj)-1;
S_in(:,In_s:In_e)=B_e(:,jj)*LO';
end
S_sum=S_in;
phi=S_in;
E=(phi)'*Q*(phi);
H=phi'*Q*A_e;

```

6. There are two iterations in the program below. Iteration i is with respect to the prediction horizon; and iteration kk is with respect to the number of input variables. Upon obtaining the convolution sum, the E and H matrices are the sum of the iterations with respect to i .

7. Continue entering the following program into the file:

```

for i=2:Np;
Eae=A_e.^i;
%calculate the finite sum S for each input
%%%%%%%%%%%%%
%For each sample i
%%%%%%%%%%%%%calculate input number 1
%specify the LO and state matrix A1
%associated with the first input
[A1,LO]=lagd(a(1),N(1));
% Laguerre function associated with input number 1
S_sum(:,1:N(1))=A_e*S_sum(:,1:N(1))+%
S_in(:,1:N(1))*(A1^(i-1))';
%move on to input number 2 and so on
In_s=1;
for kk=2:n_in;
[A1,LO]=lagd(a(kk),N(kk));
In_s=N(kk-1)+In_s;
In_e=In_s+N(kk)-1;
S_sum(:,In_s:In_e)=A_e*S_sum(:,In_s:In_e)+%
S_in(:,In_s:In_e)*(A1^(i-1))';
end
phi=S_sum;
E=E+phi'*Q*phi;
H=H+phi'*Q*Eae;
end
E=E+R_para;

```

8. This program generates the predictive control cost function for a system with an arbitrary number of inputs, states and outputs. It is a general program and will be used later on for many other purposes.

The program is very important and needs to be tested before any further applications. We test it against the results obtained using the MATLAB dlqr program.

Tutorial 3.3. Consider the mathematical model of a continuously stirred tank reactor for copolymerization of methyl methacrylate and vinyl acetate, described by the transfer function (Rao et al., 1998):

$$G(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) & g_{13}(s) & g_{14}(s) & g_{15}(s) \\ g_{21}(s) & g_{22}(s) & g_{23}(s) & g_{24}(s) & g_{25}(s) \\ g_{31}(s) & g_{32}(s) & g_{33}(s) & g_{34}(s) & g_{35}(s) \\ g_{41}(s) & g_{42}(s) & g_{43}(s) & g_{44}(s) & g_{45}(s) \end{bmatrix}, \quad (3.49)$$

where

$$\begin{aligned} g_{11} &= \frac{0.34}{0.85s + 1}; & g_{12} &= \frac{0.21}{0.42s + 1} \\ g_{13} &= \frac{0.50(0.50s + 1)}{12s^2 + 0.4s + 1}; & g_{14} &= 0 \\ g_{15} &= \frac{6.46(0.9s + 1)}{0.07s^2 + 0.3s + 1}; & g_{21} &= \frac{-0.41}{2.41s + 1} \\ g_{22} &= \frac{0.66}{1.51s + 1}; & g_{23} &= \frac{-0.3}{1.45s + 1} \\ g_{24} &= 0; & g_{25} &= \frac{-3.72}{0.8s + 1} \\ g_{31} &= \frac{0.3}{2.54s + 1}; & g_{32} &= \frac{0.49}{1.54s + 1} \\ g_{33} &= \frac{-0.71}{1.35s + 1}; & g_{34} &= \frac{-0.2}{2.72s + 1} \\ g_{35} &= \frac{-4.71}{0.008s^2 + 0.41s + 1}; & g_{41} &= 0 \\ g_{42} = g_{43} = g_{44} &= 0; & g_{45} &= \frac{1.02}{0.07s^2 + 0.31s + 1}. \end{aligned}$$

The normalized inputs into the system are the flows of monomer MMA u_1 , monomer VA u_2 , initiator u_3 , and transfer agent u_4 and the temperature of the reactor jacket u_5 . The normalized outputs of the systems are the polymer production rate y_1 , mole fraction of MMA in the polymer y_2 , average molecular weight of the polymer y_3 , and reactor temperature y_4 .

The system has five inputs and four outputs, which will test the generality of the program.

Step by Step

1. Create a new file called `testexa.m`.

2. The program below is the specification of the denominators and numerators of the transfer functions in the 5-input, 4-output system.
3. Enter the following program into the file:

```
%First row
n11=0.34;
d11=[0.85 1];
n12=0.21;
d12=[0.42 1];
n13=0.50*[0.50 1];
d13=[12 0.4 1];
n14=0;
d14=1;
n15=6.46*[0.9 1];
d15=[0.07 0.3 1];
%second row
n21=-0.41;
d21=[2.41 1];
n22=0.66;
d22=[1.51 1];
n23=-0.3;
d23=[1.45 1];
n24=0;
d24=1;
n25=-3.72;
d25=[0.8 1];
%third row
n31=0.3;
d31=[2.54 1];
n32=0.49;
d32=[1.54 1];
n33=-0.71;
d33=[1.35 1];
n34=-0.20;
d34=[2.71 1];
n35=-4.71;
d35=[0.008 0.41 1];
%fourth row
n41=0;
d41=1;
n42=0;
d42=1;
n43=0;
d43=1;
n44=0;
```

```
d44=1;
n45=1.02;
d45=[0.07 0.31 1];
```

4. The transfer function matrix for this 5 inputs and 4 outputs system is put together. After obtaining the MIMO transfer function, we obtain a minimal realization by using the command ‘ss’. Continue entering the following program

```
h=1; %sampling interval
Gs=tf({n11 n12 n13 n14 n15;n21 n22 n23 n24 n25;
n31 n32 n33 n34 n35;n41 n42 n43 n44 n45},
{d11 d12 d13 d14 d15;d21 d22 d23 d24 d25;
d31 d32 d33 d34 d35;d41 d42 d43 d44 d45});
Gsmin=ss(Gs,'min');
[Ac,Bc,Cc,Dc]=ssdata(Gsmin);
[Ap,Bp,Cp,Dp]=c2dm(Ac,Bc,Cc,Dc,h,'zoh');
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
```

5. We specify the parameters in the Laguerre functions for each input. Continue entering the following program into the file:

```
a1=0.5;
a2=0.5;
a3=0.5;
a4=0.5;
a5=0.5;
N1=10;
N2=10;
N3=10;
N4=10;
N5=10;
a=[a1 a2 a3 a4 a5];
N=[N1 N2 N3 N4 N5];
Np=100;
```

6. Augment the plant state-space model with integrators, and specify the weight matrices Q and R. Continue entering the following program into the file:

```
%%%%%%%%%%%%%%
%Augment state equations
%%%%%%%%%%%%%
A_e=eye(n1+m1,n1+m1);
A_e(1:n1,1:n1)=Ap;
A_e(n1+1:n1+m1,1:n1)=Cp*Ap;
B_e=zeros(n1+m1,n_in);
```

```
B_e(1:n1,:)=Bp;
B_e(n1+1:n1+m1,:)=Cp*Bp;
C_e=zeros(m1,n1+m1);
C_e(:,n1+1:n1+m1)=eye(m1,m1);
Q=C_e'*C_e;
R=0.1*eye(n_in,n_in);
```

7. Call function *dmpc.m* to generate Ω and Ψ matrices, based on which the feedback gain matrix K in the predictive controller is obtained using the initial condition of the Laguerre functions. Continue entering the following program into the file:

```
[Omega,Psi]=dmpc(A_e,B_e,a,N,Np,Q,R);
L_m=zeros(n_in,sum(N));
[A1,L0]=lagd(a(1),N(1));
L_m(1,1:N(1))=L0';
In_s=1;
for jj=2:n_in;
[A1,L0]=lagd(a(jj),N(jj));
In_s=N(jj-1)+In_s;
In_e=In_s+N(jj)-1;
L_m(jj,In_s:In_e)=L0';
end
K=L_m*(Omega\Psi);
Acl=A_e-B_e*K;
```

8. The above program produces the discrete-time model predictive control using Laguerre functions. The state feedback control gain is K and the closed-loop system matrix is A_{cl} .
9. Now, we will design the state feedback control using DLQR. Continue entering the program.

```
[X,Y,Z]=dlqr(A_e,B_e,Q,R);
figure
plot(Z,'ro')
hold on
plot(eig(Acl),'b*')
```

10. Change the parameter vector a and N , and see the effects of these parameters on the closed-loop eigenvalues in comparison to the DLQR solution.
11. If your plot shows that the eigenvalues of the discrete-time MPC system are almost identical to those obtained from DLQR system (see Figure 3.7), then your program has passed the test. You can retain it for future applications.

3.5.2 Predictive Control System Simulation

Tutorial 3.4. This tutorial is to produce a multivariable closed-loop simulation for predictive control systems without constraints. The performance of an unconstrained system needs to be checked before introducing the constraints. There is no observer used in the first program. An observer is included in the second program. It is useful to write up your own MATLAB simulation programs, in conjunction with your own predictive control design programs. The experience gained from the simulation will be useful in the implementation of predictive control systems.

Step by Step

1. Create a new file called *simuuc.m*.
2. The set-point signal is called *sp* with number of rows equal to the number of outputs and number of columns greater than the simulation time (N_{sim}). Enter the following program into the file:

```
function [u1,y1,deltau1,k]=
simuuc(xm,u,y,sp,Ap,Bp,Cp,N_sim,Omega,Psi,Lzerot)
%closed-loop simulation without constraints
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
Xf=[xm;(y-sp(:,1))];
for kk=1:N_sim;
eta=-(Omega\Psi)*Xf;
deltau=Lzerot*eta;
u=u+deltau;
deltau1(:,kk)=deltau;
u1(1:n_in,kk)=u;
y1(1:m1,kk)=y;
%%%%%
```

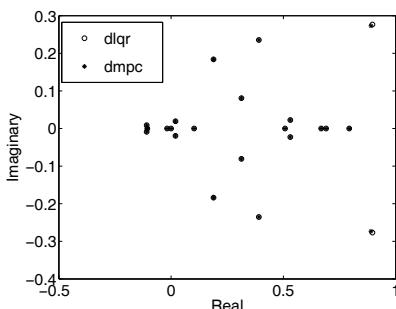
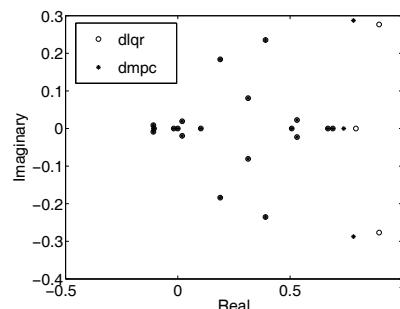
(a) Laguerre functions $a = 0.5$ (b) Pulse operator $a = 0$

Fig. 3.7. Comparison of closed-loop eigenvalues between DLQR and DMPC

```
%plant simulation
%%%%%
xm_old=xm;
xm=Ap*xm+Bp*u; % calculate xm(k+1)
y=Cp*xm; %calculate y(k+1)
%updating feedback state variable Xf
Xf=[xm-xm_old;(y-sp(:,kk+1))];
end
k=0:(N_sim-1);
```

Tutorial 3.5. We need to test the MPC simulation program *simuuc.m* before its application. The test example follows from the MPC system designed in Tutorial 3.3. This tutorial continues from MATLAB program *testexa.m*. We will next simulate closed-loop system response using the function *simuus.m*. The initial conditions for the simulation are specified first, and the function is called to produce the closed-loop responses. *Lzerot* is the matrix used to reconstruct the $\Delta u(k_i)$ with the parameter vector η (see (3.46)), which is included in the program *Mdu.m* (see Tutorial 3.7).

Step by Step

1. Copy *testexa.m* to *testexa1.m*. Continue entering the following program into *testexa1.m*.

```
y=zeros(m1,1);
u=zeros(n_in,1);
xm=zeros(n1,1);
N_sim=100;
r1=ones(1,N_sim+10);
r2=zeros(1,N_sim+10);
r3=zeros(1,N_sim+10);
r4=zeros(1,N_sim+10);
sp=[r1;r2;r3;r4];
[M,Lzerot]=Mdu(a,N,n_in,1);
[u1,y1,deltau1,k]=simuuc(xm,u,y,sp,Ap,Bp,Cp,
N_sim,Omega,Psi,Lzerot);
```

2. Plot the output signal y_1 and the control signal u_1 , which are shown in Figure 3.8.

Tutorial 3.6. This tutorial produces a closed-loop simulation for predictive control systems with an observer in the loop. The program is very similar to the program presented in Tutorial 3.4. However, the feedback variable is based on the estimated state variable $x_{\hat{h}}$.

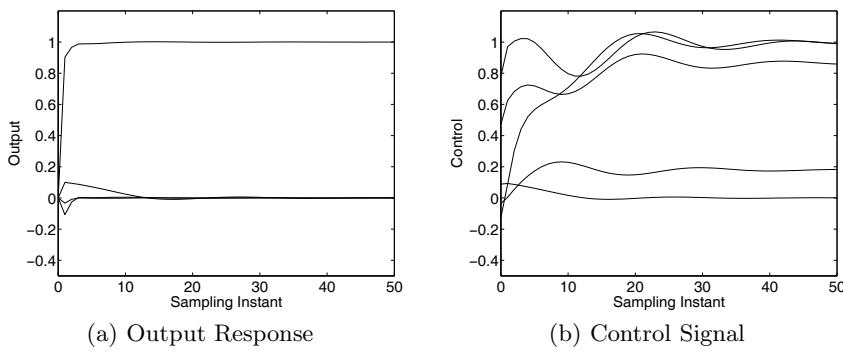


Fig. 3.8. Closed-loop simulation results

Step by Step

1. Create a new file called *simuob.m*.
 2. Enter the following program into the file:

```

function [u1,y1,deltau1,k]=simuob(xm,u,y,sp,
Ap,Bp,Cp,A,B,C,N_sim,Omega,Psi,K_ob,Lzerot)
%closed-loop simulation without constraints
[ny,n]=size(C);
[n,nu]=size(B);
X_hat=zeros(n,1);
for kk=1:N_sim;
    Xsp=[zeros(n-ny,1);sp(:,kk)];
    eta=-(Omega\Psi)*(X_hat-Xsp);
    deltau=Lzerot*eta;
    u=u+deltau; %update u
    deltau1(:,kk)=deltau;
    u1(1:nu,kk)=u; %keep u
    y1(1:ny,kk)=y; %keep y
    X_hat=A*X_hat+K_ob*(y-C*X_hat)+B*deltau;
%u and y to generate X_hat(k+1)
%%%%%
%plant simulation
%%%%%
    xm=Ap*xm+Bp*u; % calculate xm(k+1)
    y=Cp*xm; %calculate y(k+1)
end
k=0:(N_sim-1);

```

3. Notice how the set-point signal sp enters the simulation; this is exactly what we could do in the implementation. sp is a signal that has number of

rows equal to the number of outputs and number of columns greater than the number of simulation.

4. A simple modification of this program will give us the computation needed for constrained control (see Section 3.6). In the modification, we define $X_f = X_{\hat{h}} - X_{sp}$ and use *QPhild.m* to solve the constrained control problem, as

```
eta=QPhild(Omega,Psi*Xf,M,gamma);
```

where M and γ are defined through the analysis in Section 3.6. A tutorial for continuous-time constrained control is introduced in Section 7.3, which illustrates the similar steps needed for the modification.

3.6 Constrained Control Using Laguerre Functions

One of the key features of model predictive control is the ability to handle hard constraints in the design. This section will show how to systematically incorporate constraints in the design and implementation of model predictive control systems. The approach is based on application of quadratic programming method as discussed in Chapter 2.

With parameterization of the control signal trajectory using Laguerre functions, we have the flexibility to choose the locations of the future constraints. This could potentially reduce the number of constraints within the prediction horizon, and hence the on-line computational load for large-scale systems. In addition, because of the existing exponential decay factor in the Laguerre functions, the difference of the control signal is ensured to converge to zero after the transient period. Thus, it is sufficient in the majority of cases that the constraints are imposed in the transient period of the response. This in turn will reduce the number of constraints.

3.6.1 Constraints on the Difference of the Control Variable

The constrained control requires real-time optimization using quadratic programming. Assuming that state-variable information $x(k_i)$ at the sampling time k_i is given, and that the lower and upper limits on Δu are Δu^{min} and Δu^{max} , the optimization procedure is to minimize the cost function J where

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(k_i), \quad (3.50)$$

while ensuring that

$$\Delta u^{min} \leq \Delta u(k_i + m) \leq \Delta u^{max},$$

with $m = 0, 1, 2, \dots$

When using Laguerre functions in the design, the incremental control signal $\Delta u(k_i + m)$ for a single-input system is represented as

$$\Delta u(k_i + m) = L(m)^T \eta,$$

where $m = 0, 1, 2, 3, \dots$. We have the freedom to choose how many constraints should be imposed on the solution. For a multi-input system, Δu^{min} and Δu^{max} are vectors containing individual limits for each control variable. Then the constraints at a future sample m are in the matrix form

$$\Delta u^{min} \leq \begin{bmatrix} L_1(m)^T & o_2^T & \dots & o_m^T \\ o_1^T & L_2(m)^T & \dots & o_m^T \\ \vdots & \vdots & \ddots & \vdots \\ o_1^T & o_2^T & \dots & L_m(m)^T \end{bmatrix} \eta \leq \Delta u^{max}, \quad (3.51)$$

where $m = 0, 1, \dots$, denotes the set of future time instants at which we wish to impose the limits on Δu , and o_k denotes the zero vector that has the same dimension as $L_k(m)$.

Before proceeding to constrained control using Laguerre functions, we need to determine the data matrices that will be used in generating the constraints. The MATLAB tutorials below will produce functions for constrained control.

Tutorial 3.7. Write a MATLAB program for generating the matrix M used in the constraint of difference of the control signal (i.e., $\Delta u(k)$). The M matrix from this function will be part of the inequality constraints

$$\begin{aligned} M\eta &\leq \Delta U^{max} \\ -M\eta &\leq -\Delta U^{min} \end{aligned}$$

where ΔU^{max} contains upper limits for Δu and ΔU^{min} contains the lower limits for Δu . We assume that the number of inputs is denoted by n_{in} , the number of future samples for constraints to be imposed is N_c , and the poles of the Laguerre functions are $a = [a_1 \ a_2 \ \dots]$ and the number of terms is denoted by $N = [N_1 \ N_2 \ \dots]$. The program will be a general formulation of the constraints imposed on all N_c samples.

Step by Step

1. Create a new file called *Mdu.m*.
2. M is the data matrix for imposing constraints on the first N_c samples on $\Delta u(k_i)$. The block matrix *Lzerot* is used for constructing the control signal.
3. Enter the following program into the file:

```
function [M,Lzerot]=Mdu(a,N,n_in,Nc)
%a and N are for the Laguerre functions
%n_in is the number of inputs
%Nc is the number of constraints
N_pa=sum(N);
M=zeros(n_in,N_pa);
```

```

M_du1=zeros(n_in,N_pa);
k0=1;
[A1,L0]=lagd(a(k0),N(k0));
M_du1(1,1:N(1))=L0';
cc=N(1);
for k0=2:n_in;
    [A1,L0]=lagd(a(k0),N(k0));
    M_du1(k0,cc+1:cc+N(k0))=L0';
    cc=cc+N(k0);
end
Lzerot=M_du1;
M=M_du1;

```

4. The rest of the blocks are constructed iteratively. Enter the following program into the file.

```

for kk=2:Nc
    k0=1;
    [A1,L0]=lagd(a(k0),N(k0));
    L=A1^(kk-1)*L0;
    M_du1(1,1:N(1))=L';
    cc=N(1);
    for k0=2:n_in;
        [A1,L0]=lagd(a(k0),N(k0));
        L=A1^(kk-1)*L0;
        M_du1(k0,cc+1:cc+N(k0))=L';
        cc=cc+N(k0);
    end
    M=[M;M_du1];
end

```

5. Test the program using $a = 0$, $N = 3$, $n_{in} = 1$ and $N_c = 3$; then $a = [0.2 \ 0.3]$, $N = [3 \ 3]$, $n_{in} = 2$, $N_c = 5$.

Example 3.6. Suppose that a continuous-time system is described by the transfer function

$$G(s) = \frac{1}{s^2 + 2\xi\omega s + \omega^2},$$

where $\xi = 0.1$ and $\omega = 3$. Suppose that the sampling interval $\Delta t = 0.1$, and the weight matrices are $Q = C^T C$ with $C = [0 \ 0 \ 1]$ and $R = 0.3$. The prediction horizon is chosen as $N_p = 46$. We assume that at $k_i = 10$, $x(k_i) = [0.1 \ 0.2 \ 0.3]^T$. Within one optimization window, find the optimal η and $\Delta u(k_i + m)$ subject to the constraints

$$-1 \leq \Delta u(k_i + m) \leq 0.25.$$

Solution. Because this is a severely under-damped system, the optimal control trajectory is oscillatory. Let us choose $a = 0.7$ and $N = 8$. Figure 3.9

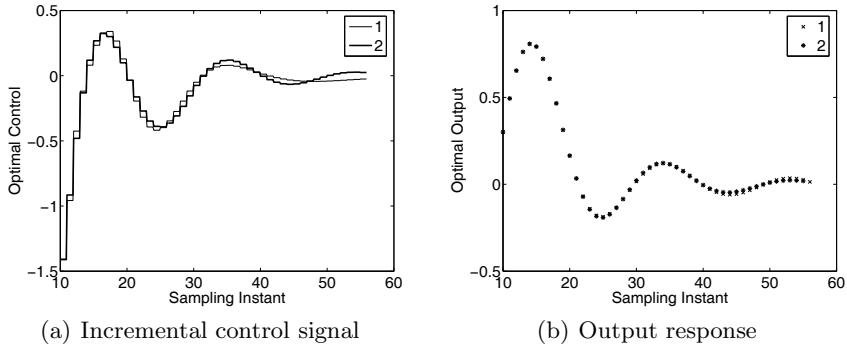


Fig. 3.9. Optimal solution without constraints. Key: line (1) DMPC; line (2) DLQR

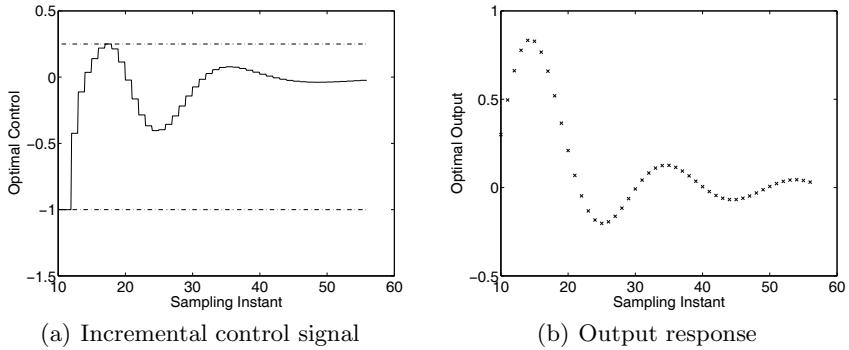
shows the comparison between the control signal and output response between Laguerre-based DMPC and DLQR. It is seen from this figure that with this choice of a and N , without constraints, the responses are almost identical. Incidentally, if the parameter $a = 0$, which corresponds to the traditional approach, then N is required to be approximately 40 in order to achieve similar performance. To ensure that the constraints are satisfied for the whole response, the first 15 samples of Δu are put into the constrained solution. This gives 30 inequalities to be imposed. Because the data matrices have large dimension, the MATLAB script below is used to illustrate the procedure.

```
[Omega,Psi]=dmpc(A,B,a,N,Np,Q,R);
Deltau_min=-1;
Deltau_max=0.25;
Nc=15;
X0=[0.1;0.2;0.3];
[M0,Lzerot]=Mdu(a,N,1,Nc);
M=[M0;-M0];
gamma=[Deltau_max*ones(Nc,1);-Deltau_min*ones(Nc,1)];
eta=QPhild(Omega,Psi*X0,M,gamma);
```

Figure 3.10 shows the constrained results within one optimization window. Although there are 30 constraints imposed, there are only three active constraints, which can be found through visual inspection of Figure 3.10a. The first two samples and the 17th sample are the activated constraints in the solution. By comparing Figure 3.10 with Figure 3.9 it can be seen that activated constraints resulted in little performance deterioration.

3.6.2 Constraints on the Amplitudes of the Control Signal

Suppose that the limits on the control signals are u^{\min} and u^{\max} . Noting that the increment of the control signal is $u(k) = \sum_{i=0}^{k-1} \Delta u(i)$, then the inequality

**Fig. 3.10.** Optimal solution with constraints

constraint for the future time k , $k = 1, 2, \dots$, is expressed as

$$u^{min} \leq \begin{bmatrix} \sum_{i=0}^{k-1} L_1(i)^T & o_2^T & \dots & o_m^T \\ o_1^T & \sum_{i=0}^{k-1} L_2(i)^T & \dots & o_m^T \\ \vdots & \vdots & \vdots & \vdots \\ o_1^T & o_2^T & \dots & \sum_{i=0}^{k-1} L_m(i)^T \end{bmatrix} \eta + u(k_i - 1) \\ \leq u^{max}, \quad (3.52)$$

where $u(k_i - 1)$ is the previous control signal, and o_k^T is a zero row vector with the same dimension as $L_k(0)^T$.

Tutorial 3.8. Write a MATLAB program for generating the matrix M used in the constraint of the control signal (i.e., $u(k)$). The M matrix from this function will be part of the inequality constraints

$$\begin{aligned} M\eta &\leq U^{max} - \bar{u}(k_i - 1) \\ -M\eta &\leq -U^{min} + \bar{u}(k_i - 1) \end{aligned}$$

where $\bar{u}(k_i - 1)$ is the vector containing the past u value. We assume that the number of input is denoted by n_{in} , the number of future samples for constraints to be imposed is N_c , and the poles of the Laguerre functions are $a = [a_1 \ a_2 \ \dots]$ and the number of terms is denoted by $N = [N_1 \ N_2 \ \dots]$. The program will be a general formulation of the constraints imposed on all N_c samples. The program is written on the basis of Tutorial 3.7 by adding the Laguerre vector matrix recursively. Since the control amplitude constraints are the most frequently imposed constraints, it is important that we produce the matrix correctly.

Step by Step

1. Create a new file called *Mu.m*.

2. Enter the following program into the file:

```

function M=Mu(a,N,n_in,Nc)
%a and N are for the Laguerre functions
%n_in is the number of inputs
%Nc is the number of constraints
N_pa=sum(N);
M=zeros(n_in,N_pa);
M_du1=zeros(n_in,N_pa);
k0=1;
[A1,L0]=lagd(a(k0),N(k0));
M_du1(1,1:N(1))=L0';
cc=N(1);
for k0=2:n_in;
[A1,L0]=lagd(a(k0),N(k0));
M_du1(k0,cc+1:cc+N(k0))=L0';
cc=cc+N(k0);
end
M=M_du1;
Ms=M_du1;
for kk=2:Nc
k0=1;
[A1,L0]=lagd(a(k0),N(k0));
L=A1^(kk-1)*L0;
M_du1(1,1:N(1))=L';
cc=N(1);
for k0=2:n_in;
[A1,L0]=lagd(a(k0),N(k0));
L=A1^(kk-1)*L0;
M_du1(k0,cc+1:cc+N(k0))=L';
cc=cc+N(k0);
end
Ms=Ms+M_du1;
M=[M;Ms];
end

```

3. Test the program using $a = 0$, $N = 3$, $n_{in} = 1$ and $N_c = 3$.

Example 3.7. We assume the same system in Example 3.6, which is

$$G(s) = \frac{1}{s^2 + 2\xi\omega s + \omega^2},$$

where $\xi = 0.1$ and $\omega = 3$. Suppose that the sampling interval $\Delta t = 0.1$, and the weight matrices are $Q = C^T C$ with $C = [0 \ 0 \ 1]$ and $R = 0.3$. The prediction horizon is chosen as $N_p = 46$. We assume that at $k_i = 10$, $x(k_i) = [0.1 \ 0.2 \ 0.3]^T$. Find the optimal η and $\Delta u(k_i + m)$ subject to the

constraints on the control signal u , with the assumption at $k_i - 1$, $u(k_i - 1) = 6$. The constraints are imposed for all future samples of the control signal such that

$$1.8 \leq u(k_i + m) \leq 4$$

where $m = 0, 1, 2, \dots, 46$.

Solution. In order to impose constraints on all future control movement, there are 92 (46×2) inequality constraints involved. The following MATLAB script is used to generate the constraints and the optimal solution with respect to the constraints.

```
[Omega,Psi]=dmpc(A,B,a,N,Np,Q,R);
X0=[0.1;0.2;0.3];
up=6;
u_min=1.8;
u_max=4;
M0=Mu(a,N,1,Np);
M=[M0;-M0];
gamma=[(u_max-up)*ones(Nc,1);(-u_min+up)*ones(Nc,1)];
eta=QPhild(Omega,Psi*X0,M,gamma);
```

There are four active constraints as shown in Figure 3.11b. All the constraints are satisfied.

Example 3.8. Following from Examples 3.6 and 3.7, at the first sample, with given initial condition $u(k_i - 1) = 6$, the constraints on $u^{\min} \geq 1.8$ and $\Delta u^{\min} \geq -1$ both became activated at the first sample. However, they have both worked in separate examples. This example shows that when both constraints are put together, they cannot be met at the same initial sample. As a result, the optimal solution is compromised.

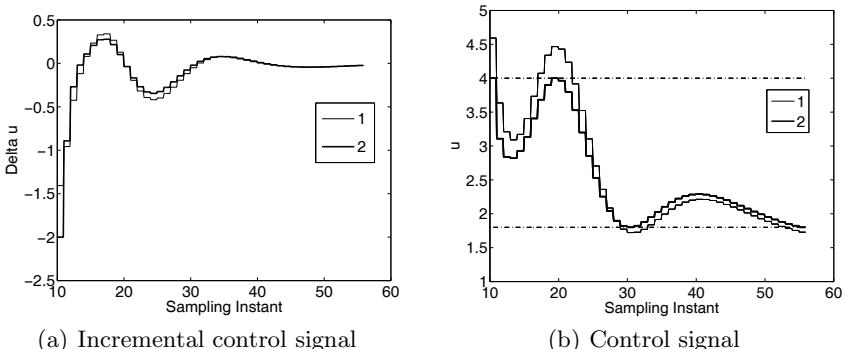


Fig. 3.11. Optimal solution with constraints on u . Key: line (1) without constraints; line (2) with constraints

Solution. When all the constraints are imposed on Δu and u for $N_p = 48$, the number of constraints is $4 \times N_p = 192$. The following MATLAB script is used to generate the constraints and find the optimal solution. A , B , a , N , N_p , Q , and R are specified. Here we only show how the constraints are formulated and applied.

```
[Omega,Psi]=dmpc(A,B,a,N,Np,Q,R);
X0=[0.1;0.2;0.3];
up=6;
%%%%%
Deltau_min=-1;
Deltau_max=0.25;
u_min=1.8;
u_max=4;
Nc=Np;
[M1,Lzerot]=Mdu(a,N,1,Nc);
M0=Mu(a,N,1,Nc);
M=[M0;-M0;M1;-M1];
gamma=[(u_max-up)*ones(Nc,1);(-u_min+up)*ones(Nc,1);
Deltau_max*ones(Nc,1); -Deltau_min*ones(Nc,1)];
eta=QPhild(Omega,Psi*X0,M,gamma);
```

Figure 3.12 shows the incremental control signal (Δu), the control signal (u) and output signal y in comparison with the unconstrained responses. It is seen from Figure 3.12b that some of the constraints on the control signal were not met. By studying the multipliers, it is noticed that there were eight active constraints and the multipliers could not converge because the active constraints were in conflict. The values of the positive multipliers are listed after 88 iterations in the calculation as:

$$\lambda_{act} = [34.2944 \quad 0.0545 \quad 0.0261 \quad 0.0654 \quad 0.0240 \quad 0.0474 \quad 34.4705 \quad 0.0136]$$

Numerically, the M_{act} matrix has row dependence, and as a result, the constraints are compromised (see Figure 3.12).

One comment relates to the number of constraints used in this example. Although there are 192 constraints used in the optimization, there are only 8 active constraints. The rest of the constraints are inactive and have no effect on the optimal solution. Because of receding horizon control, where the first sample of the optimal control $\Delta u(k_i)$ ($\Delta u(k_i) = L(0)^T \eta = -1$) is implemented, so from this point of view, only the constraint on the first sample had an effect in this example.

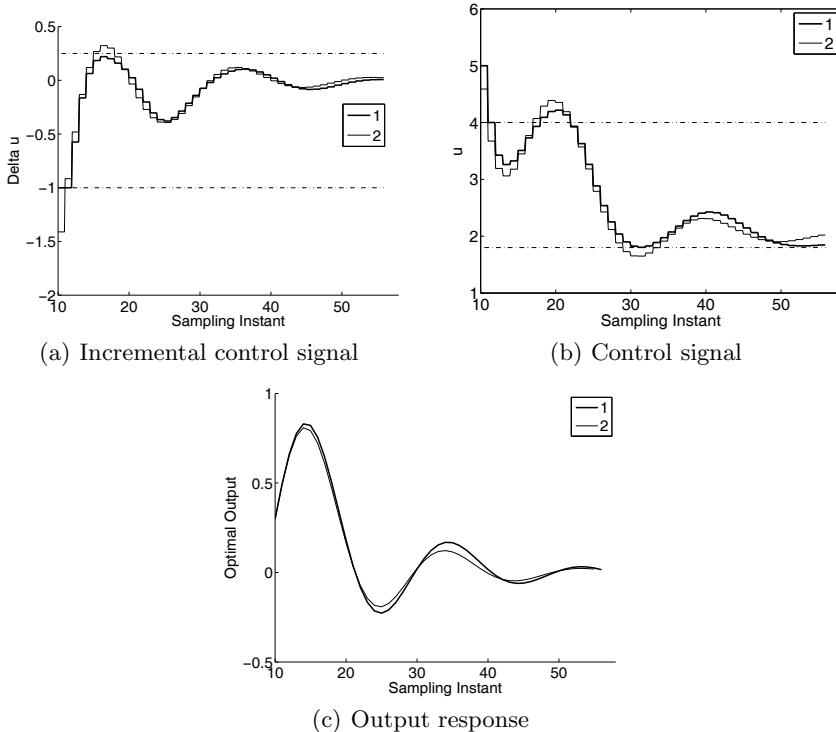


Fig. 3.12. Optimal solution with constraints on u and Δu . Key: Line (1) with constraints; Line (2) without constraints

Example 3.9. This example shows that by changing the specification of the constraints on Δu as

$$-2 \leq \Delta u(k) \leq 0.25$$

while the constraints on u remain the same: $1.8 \leq u(k) \leq 4$, all the constraints are satisfied.

Solution. By a quick calculation, with given initial condition $u(9) = 6$, the constraints at the first sample should be satisfied. With the same MATLAB script used in Example 3.8, except that the minimum of $\Delta u(k)$ is -2 , instead of -1 , the Lagrange multipliers converged to the set

$$[0.2307 \ 0.0549 \ 0.0242 \ 0.0654 \ 0.0268 \ 0.0480 \ 0.0112].$$

There are seven active constraints. Figure 3.13 shows the results of constrained control in comparison with the case of unconstrained control. It is seen that all the constraints are satisfied.

These examples raise the question of how many constraints should be included in the optimization problem. Because the Laguerre functions are exponentially

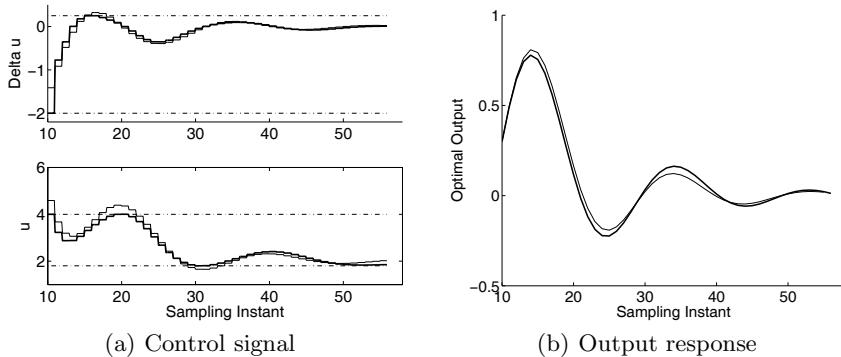


Fig. 3.13. Optimal solution with constraints on Δu and u . Key: solid-line without constraints; darker-solid-line with constraints

decaying, the constraints on $\Delta u(k_i + j)$ will be automatically satisfied after some initial period. Consequently, it is usual to impose constraints on the first few samples of $\Delta u(k_i + j)$ and $u(k_i + j)$.

When controlling a MIMO system with constraints, it is usual to impose constraints on the first samples of $\Delta u(k_i + j)$ and $u(k_i + j)$ because the number of constraints will increase rapidly in this situation.

3.7 Stability Analysis

When the constraints in model predictive control are activated, the control law effectively becomes a nonlinear control problem. Thus, the stability properties of linear time-invariant systems do not apply. However, a remarkable property of model predictive control is that one can establish stability of the closed-loop system under certain conditions. In this section, we will first find out the stabilizing conditions and then explore the links between the conditions and the design parameters.

There is a rich literature on the topic of stability of predictive control systems (see Keerthi and Gilbert, 1988, Mayne *et al.*, 2000, Bitmead *et al.*, 1990, Rawlings and Muske, 1993, Muske and Rawlings, 1993).

3.7.1 Stability with Terminal-State Constraints

Recall that the model predictive control is established using the principle of receding horizon control. That is, at current sample time k_i the future of the control trajectory $\Delta u(k_i + m)$, $m = 0, 1, 2, \dots, N_p$ is optimized by minimizing the cost function

$$J = \sum_{m=1}^{N_p} x(k_i + m | k_i)^T Q x(k_i + m | k_i) + \sum_{m=0}^{N_p-1} \Delta u(k_i + m)^T R \Delta u(k_i + m), \quad (3.53)$$

subject to constraints, and upon obtaining the optimal control sequence, only the first sample $\Delta u(k_i)$ is implemented. At the next sample time $k_i + 1$, the same optimization and implementation procedures are repeated. The core to establishing closed-loop stability is based on an equality constraint on the terminal state, which is $x(k_i + N_p | k_i) = 0$. For notational simplicity, we replace k_i by k in this section. We also consider a single-input system.

Theorem 3.1. *We assume that*

1. *an additional constraint is placed on the final state of the receding horizon optimization problem: $x(k + N_p | k) = 0$, where $x(k + N_p | k)$ is the terminal state resulting from the control sequence $\Delta u(k + m) = L(m)^T \eta$, $m = 0, 1, 2, \dots, N_p$; and*
2. *for each sampling instant, k , there exists a solution η such that the cost function J is minimized subject to the inequality constraints and terminal equality constraint $x(k + N_p | k) = 0$.*

Subject to the assumptions, the closed-loop model predictive control system is asymptotically stable.

Proof. We consider the cost function (3.53). The key to the stability result is to construct a Lyapunov function for the model predictive control system.

Choose the Lyapunov function $V(x(k), k)$ as the minimum of the finite horizon cost function (J_{min})

$$V(x(k), k) = \sum_{m=1}^{N_p} x(k+m | k)^T Q x(k+m | k) + \sum_{m=0}^{N_p-1} \Delta u(k+m)^T R \Delta u(k+m), \quad (3.54)$$

where $x(k+m | k) = A^m x(k) + \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \eta^k$ and η^k is, at time k , the parameter vector solution of the original cost function (3.53) with respect to both inequality and equality constraints, and $\Delta u(k+m) = L(m)^T \eta^k$. The existence of η^k is ensured by the second assumption stated in the theorem. Namely, $V(x(k), k) = J_{min}$, where η^k is a function of $x(k)$.

It is seen that $V(x(k), k)$ is positive definite and $V(x(k), k)$ tends to infinity if $x(k)$ tends to infinity. Similarly, at time $k + 1$, the Lyapunov function becomes

$$\begin{aligned} V(x(k+1), k+1) &= \sum_{m=1}^{N_p} x(k+1+m | k+1)^T Q x(k+1+m | k+1) \\ &\quad + \sum_{m=0}^{N_p-1} \Delta u(k+1+m)^T R \Delta u(k+1+m), \end{aligned} \quad (3.55)$$

and $x(k+1+m \mid k+1) = A^m x(k+1) + \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \eta^{k+1}$, η^{k+1} is the parameter vector solution at $k+1$ and $\Delta u(k+1+m) = L(m)^T \eta^{k+1}$.

We need to make a link between the Lyapunov function at sample time $k+1$ and the function at sample time k . Assuming that all constraints are satisfied at the sample time k , a feasible solution of η^{k+1} (not the optimal one) for the initial state information $x(k+1)$ in the receding horizon is η^k given that $x(k+1)$ is the response one step ahead of $x(k)$ related to $x(k)$ by

$$x(k+1) = Ax(k) + B\Delta u(k).$$

Therefore, the feasible control sequence at $k+1$ is to shift the elements in $L(0)^T \eta^k, L(1)^T \eta^k, L(2)^T \eta^k, \dots, L(N_p - 1)^T \eta^k$ one step forward and replace the last element by zero to obtain the sequence $L(1)^T \eta^k, L(2)^T \eta^k, \dots, L(N_p - 1)^T \eta^k, 0$. Because of optimality in the solution of η^{k+1} , it is seen that

$$V(x(k+1), k+1) \leq \bar{V}(x(k+1), k+1), \quad (3.56)$$

where $\bar{V}(x(k+1), k+1)$ is similar to (3.55) except that the control sequence is replaced by the feasible sequence $L(1)^T \eta^k, L(2)^T \eta^k, \dots, L(N_p - 1)^T \eta^k, 0$. The difference between $V(x(k+1), k+1)$ and $V(x(k), k)$ is then bounded by

$$V(x(k+1), k+1) - V(x(k), k) \leq \bar{V}(x(k+1), k+1) - V(x(k), k). \quad (3.57)$$

Note that because the $\bar{V}(x(k+1), k+1)$ shares the same control sequence and the same state sequence with $V(x(k), k)$ for the sample time $k+1, k+2, \dots, k+N_p - 1$, the difference between these two functions is

$$\begin{aligned} \bar{V}(x(k+1, k+1) - V(x(k), k) &= x(k+N_p \mid k)^T Q x(k+N_p \mid k) \\ &\quad - x(k+1)^T Q x(k+1) - \Delta u(k)^T R \Delta u(k). \end{aligned} \quad (3.58)$$

From the first assumption, we have

$$\bar{V}(x(k+1, k+1) - V(x(k), k) = -x(k+1)^T Q x(k+1) - \Delta u(k)^T R \Delta u(k). \quad (3.59)$$

Hence, the difference of the Lyapunov function is

$$V(x(k+1), k+1) - V(x(k), k) \leq -x(k+1)^T Q x(k+1) - \Delta u(k)^T R \Delta u(k) < 0, \quad (3.60)$$

which we see is negative. Hence, we have established the asymptotic stability of the model predictive control system.

3.7.2 Stability with Large Prediction Horizon

The requirement of terminal state constraint $x(k+N_p \mid k) = 0$ was originally from (3.58). However, as we know from Chapter 2, equality constraints are active constraints. Therefore, in order to have constrained optimal solutions, firstly, the number of decision variables should be larger than the number of

active constraints. Thus, having the terminal-state constraints, the number of decision variables, *i.e.*, the dimension of η , is likely to be increased. Secondly, there is a possibility that the active constraints from the terminal-state could cause a linear dependence with other inequality constraints such as constraints on input and output signals. When this happens, the active constraints (including the terminal-state constraints) may not be satisfied. For these reasons, the terminal-state constraints are seldom used in practice. Instead, we explore under what conditions the term $x(k + N_p | k)^T Q x(k + N_p | k)$ is sufficiently small so that the difference of the Lyapunov function is guaranteed to be negative.

From the study in Section 3.3.6, without constraints, we understand that for a large prediction horizon and a large N in the Laguerre functions, the optimal control trajectory $\Delta u(k + m)$ converges to the underlying optimal control dictated by a discrete-time linear quadratic regulator with the same weight matrices Q and R . The cost functions of predictive control and DLQR become nearly identical. We illustrated this by several examples (see Examples 3.2 to 3.5). When this happens, because of the asymptotic stability of the DLQR system, with a large N , for some large prediction horizon N_p ,

$$x(k + N_p | k)^T Q x(k + N_p | k) \approx 0$$

Consequently, this leads to the closed-loop stability of the predictive control system from the Lyapunov function analysis.

However, we also note that the plants used in these examples are stable systems, and the underlying optimal DLQR control trajectory converges to zero in less than 20 samples. In other words, we have achieved the convergence without using a very large prediction horizon in the computation, but sufficiently covered the transient period of the state response, and consequently avoided a numerical ill-conditioning problem that will be discussed in Chapter 4.

Since we do not deploy the equality terminal-state constraint in the design of MPC, how does the predictive controller stabilize the plant when constraints become active? The stabilization is argued through the combination of dual mode control system and a terminal-state constraint set (Mayne *et al*, 2000). The terminal-state equality constraint, $x(k + N_p | k) = 0$, is considered to be severe as we argued before. Thus a relaxation is to specify a terminal constraint set X_0 , that contains the origin. The idea of a terminal constraint set was introduced by Michalska and Mayne (1993). It is assumed that within the terminal constraint set X_0 the constraints are inactive and conventional state feedback control is adequate when the state is within this terminal set. With the idea of terminal set, in order to obtain stability, a predictive controller with constraints is used first to drive the state into such a set and then switch to another control law that will stabilize the system with the initial conditions equal to the terminal state conditions of the predictive control. This is in the spirit of classical dual-mode control systems (Anderson and Moore, 1971).

Here, stability of the predictive control systems will be examined from a combination of dual mode control system and a terminal-state constraint set. The key to establish stability is to define the state feedback control law that will not violate the constraints and will keep the state in the set, once the state enters the terminal set.

In Examples 3.7 to 3.9, we have examined the optimal control law within one optimization window. It is seen that the predictive control law is identical to the underlying optimal LQR when constraints are not active. When constraints are active, the predictive control law satisfies the constraints at the places where they are violated, and yet the control trajectory returns to the underlying optimal control after the constraints are not active. Therefore, the state feedback control law within the terminal-state set is the original optimal control law with controller K_{lqr} , defined by

$$\Delta u(k_i + m) = -K_{lqr}x(k_i + m \mid k_i),$$

where $m = N_0, N_0 + 1, N_0 + 2, \dots$ with N_0 defining the sample time when the state feedback control law becomes effective. We can actually identify the time N_0 when we compare the control trajectory $\Delta u(\cdot)$ in the presence of constraints with the one without constraints. For instance, in Examples 3.8 and 3.9, we can see after the first 33 samples ($N_0 = 33$), the predictive control trajectory returns to the LQR control trajectory, all the constraints are satisfied and the state is regulated to zero (see Figures 3.12a and 3.13). Also it is interesting to note that in Example 3.8, there are conflict constraints close to the sample time 20 (see Figure 3.12). However, the predictive control solution is compromised to reach a sub-optimal solution at the conflicting point and returns to the optimal LQR trajectory after the initial period.

Further analysis of closed-loop stability is given in Chapter 4, where an asymptotically stable model is used in the design via exponential data weighting.

3.8 Closed-form Solution of Constrained Control for SISO Systems

The question we often ask as an engineer is whether there is a simplified solution to a complex problem. Simplicity is often the key that leads to a wider range of applications. The quadratic programming approach is certainly needed for a multi-input and multi-output system, as we note that it is an effective way to identify the active constraints when there are many variables interacting with each other. As for a single-input and single-output system, the variables are limited to control $u(k)$, and difference of the control $\Delta u(k)$ and output $y(k)$ signals. It is possible to find analytical solutions for the constrained control problem. Also, the role of observer can be eliminated to simplify the design. In the traditional approach to MPC, constraints

are introduced on the variables in the optimization window. As a result, the number of constraints rapidly grows even for a single-input and single-out system. Certainly in the presence of a large number of constraints, quadratic programming is the most effective way to sort out the active constraints. The question here is whether this is necessary. If not, what might be compromised? To answer this question, we return to the receding horizon control principle, which states that although the optimal control is sought for the whole trajectory within the optimization window, only the first sample of the trajectory is implemented. Therefore, if the first sample of the constraint is an active constraint, whether it is based on Δu or u , then the optimal solution is obtained based on the active constraint that becomes an equality constraint, and the rest of constraints do not affect the optimal solution. The rest of the constraints will affect the optimal solution if they are activated, and the first sample of the constraint is not active. From a practical point of view, it is possible only to impose the constraints on the first sample of the variables in the optimization window. The rest of the constraints are neglected on the assumption that they are not active or they have a small effect on the closed-loop performance. Based on this rationale, there are six constraints for a SISO system, which are the upper and lower limits of the control variable, difference of the control variable and output variable. The closed-form solutions for the six cases are discussed as follows.

Closed-form Solution with Constraints on Difference of the Control Variable

Suppose that the constraints are expressed as $\Delta u^{min} \leq \Delta u(k) \leq \Delta u^{max}$. At sample instant k_i , the state variable $x(k_i)$ is available that also contains set-point information for simplicity, minimization of the cost function

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(k_i),$$

subject to the constraints on $\Delta u(k_i) (= L(0)^T \eta)$ leads to three possible solutions:

1. $\eta = -\Omega^{-1} \Psi x(k_i)$, if $\Delta u^{min} < L(0)^T \eta < \Delta u^{max}$. The optimal solution is the global optimal solution that minimizes the quadratic cost function J .
2. If $L(0)^T \eta \leq \Delta u^{min}$, where $\eta = -\Omega^{-1} \Psi x(k_i)$, then the constrained optimal solution η is the one that leads to $\Delta u(k_i) = \Delta u^{min}$. This is because this constraint is the active constraint and the active constraint becomes an equality constraint. Hence the optimal solution satisfies the equality, which is imposed by Δu^{min} . Instead of finding out the value of η , we directly take $\Delta u(k_i) = \Delta u^{min}$ as the optimal solution.
3. Similarly, when $L(0)^T \eta \geq \Delta u^{max}$ with $\eta = -\Omega^{-1} \Psi x(k_i)$, this constraint becomes active. The optimal solution η leads to the one that $\Delta u(k_i) = \Delta u^{max}$.

After finding $\Delta u(k_i)$, as before, the optimal control is $u(k_i) = u(k_i - 1) + \Delta u(k_i)$.

Closed-form Solution with Constraints on Control Variable

Assume that the constraints on the control variable are expressed as $u^{min} \leq u(k) \leq u^{max}$. Suppose that at sample time k_i , the previous control signal is $u(k_i - 1)$. There are, again, three different cases:

1. $\eta = -\Omega^{-1}\Psi x(k_i)$ if $u^{min} < u(k_i - 1) + L(0)^T \eta < u^{max}$, which says that if the global optimal solution satisfies the constraints, then the optimal solution is the global optimal solution.
2. If $u(k_i - 1) + L(0)^T \eta \leq u^{min}$, where $\eta = -\Omega^{-1}\Psi x(k_i)$, then $u(k_i) = u^{min}$, from which we derive $\Delta u(k_i) = u^{min} - u(k_i - 1)$.
3. Similarly, if $u(k_i - 1) + L(0)^T \eta \geq u^{max}$, where $\eta = -\Omega^{-1}\Psi x(k_i)$, then $u(k_i) = u^{max}$, and $\Delta u(k_i) = u^{max} - u(k_i - 1)$.

The information about $\Delta u(k_i)$ calculated using the limits is important, because it will be updated in the implementation of the control as a feedback variable.

Closed-form Solution with Constraints on the Output Variable

Assume that the predictive control is designed using the state-space model with (A, B, C) matrices and the constraints on the output are expressed as $y^{min} \leq y(k) \leq y^{max}$. At sample time k_i , the prediction of $x(k_i + 1 | k_i)$ is described by the relation

$$x(k_i + 1 | k_i) = Ax(k_i) + BL(0)^T \eta, \quad (3.61)$$

which gives the predicted output as

$$y(k_i + 1 | k_i) = CAx(k_i) + CBL(0)^T \eta. \quad (3.62)$$

There are three possible solutions for this case.

1. At sample time k_i with given plant information $x(k_i)$, if

$$y^{min} < CAx(k_i) + CBL(0)^T \eta < y^{max}, \quad (3.63)$$

where $\eta = -\Omega^{-1}\Psi x(k_i)$. Then the optimal solution η is the solution with the constraints.

2. If the constraint on the lower limit is violated such that

$$CAx(k_i) + CBL(0)^T \eta \leq y^{min}, \quad (3.64)$$

where $\eta = -\Omega^{-1}\Psi x(k_i)$, then this constraint becomes active. The constrained optimal solution is not as straightforward as the case for the constraints on the control signal, however, an analytical solution is found by treating this constraint as an equality constraint. Namely, we need to find η that will minimize the cost function J as well as satisfy the inequality below:

$$-CBL(0)^T \eta \leq -y^{min} + CAx(k_i). \quad (3.65)$$

Defining $M_{act} = -CBL(0)^T$ for notational simplicity, the Lagrange multiplier λ_{act} has an analytical expression:

$$\lambda_{act} = -(M_{act}\Omega^{-1}M_{act}^T)^{-1}(-y^{min} + CAx(k_i) + M_{act}\Omega^{-1}\Psi x(k_i)). \quad (3.66)$$

Based on this expression, the analytical optimal solution of the parameter vector η is obtained as

$$\eta = -\Omega^{-1}(\Psi x(k_i) + M_{act}^T \lambda_{act}). \quad (3.67)$$

3. Similarly, if the constraint on the upper limit is violated when $\eta = -\Omega^{-1}\Psi x(k_i)$, then the constrained optimal solution η is the one that will minimize the cost function J and satisfy the inequality:

$$CBL(0)^T \eta \leq y^{max} - CAx(k_i). \quad (3.68)$$

In this case with $M_{act} = CBL(0)^T$, the analytical solution for the Lagrange multiplier is expressed as

$$\lambda_{act} = -(M_{act}\Omega^{-1}M_{act}^T)^{-1}(y^{max} - CAx(k_i) + M_{act}\Omega^{-1}\Psi x(k_i)), \quad (3.69)$$

which gives the constrained optimal solution η by (3.67).

Ranking of the Constraints

What if the constraints on the difference of the control, the control and the output were all violated or any combinations of two kinds of constraints were violated? If they were violated at the same time, could the violated constraints be treated as active constraints so that the optimal control satisfies the constraints? The answer is negative. At any given sample time k_i and state variable $x(k_i)$, the constrained optimal solution η exists under the condition that only one constraint is active. This observation is based on the fact that the three types of constraints are linearly dependent. For instance, the $M_{act} = \pm L(0)^T$ for the constraints on Δu ; and for the constraints on u , $M_{act} = \pm L(0)^T \Delta t$ and for the constraints on y , $M_{act} = \pm CBL(0)^T$. Therefore, when putting them together, the determinant of $(M_{act}\Omega^{-1}M_{act}^T)$ is zero and there is no solution for λ_{act} . This leads to the ranking of the constraints for their importance so that the most crucial constraint will be satisfied, and the less important constraints will be left alone.

The most important constraints are associated with the control signals, whether it is the amplitude of the control or the difference of the control. The less important constraints are the output constraints, because the effectiveness of this type of constraints is dependent on the existence of an accurate model for prediction. This is evident from the prediction (3.62) where the model is used to predict the next step of y . In addition, if an observer is used, $x(k_i)$

is replaced by the estimated state $\hat{x}(k_i)$, which leads to the question of the effect of estimation error on the constraints.

After ranking the constraints, assuming that the most important constraints are on the amplitude of the control, and the less important constraints are on the difference of the control and the least important constraints are on the output, the implementation is as follows:

1. The least important constraints on y if they are violated, will be treated as an active constraint so as to obtain η that will satisfy this constraint on y .
2. Check whether this η will lead to the satisfaction of the constraints on Δu . If not, Δu will be modified to satisfy the constraints on Δu . With this new Δu , we need to check if the constraints on u are satisfied. If not, u will be modified to satisfy the constraints on u .

3.8.1 MATLAB Tutorial: Constrained Control of DC Motor

A DC motor has a continuous-time transfer function as

$$G(s) = \frac{K}{s(Ts + 1)}, \quad (3.70)$$

where the control signal is input voltage to the motor and the output is the angular position. The discrete-time model with zero-order-hold is expressed as

$$G(z) = \frac{b_1 z + b_2}{z^2 + a_1 z + a_2}. \quad (3.71)$$

Instead of using an observer in the implementation, we will choose the state variables according to the input and output signals so that the state variables are directly measurable. To do so, let us look at the difference equation that relates the input to the output:

$$y(k+2) + a_1 y(k+1) + a_2 y(k) = b_1 u(k+1) + b_2 u(k). \quad (3.72)$$

Choosing the state variable as

$$x_m(k) = [y(k) \ y(k-1) \ u(k-1)]^T$$

$$\begin{bmatrix} y(k+1) \\ y(k) \\ u(k) \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & b_2 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y(k) \\ y(k-1) \\ u(k-1) \end{bmatrix} + \begin{bmatrix} b_1 \\ 0 \\ 1 \end{bmatrix} u(k)$$

$$y(k) = [1 \ 0 \ 0] x_m(k). \quad (3.73)$$

The augmented state-space model is obtained by choosing the state variable vector as

$$x(k) = [\Delta y(k) \ \Delta y(k-1) \ \Delta u(k-1) \ y(k)]^T,$$

leading to

$$\begin{bmatrix} \Delta y(k+1) \\ \Delta y(k) \\ \Delta u(k) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & b_2 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -a_1 & -a_2 & b_2 & 1 \end{bmatrix} \begin{bmatrix} \Delta y(k) \\ \Delta y(k-1) \\ \Delta u(k-1) \\ y(k) \end{bmatrix} + \begin{bmatrix} b_1 \\ 0 \\ 1 \\ b_1 \end{bmatrix} \Delta u(k)$$

$$y(k) = [0 \ 0 \ 0 \ 1] x(k). \quad (3.74)$$

To be more precise, here $\Delta y(k) = y(k) - y(k-1)$ and $\Delta u(k) = u(k) - u(k-1)$. At a sampling instant k_i , with plant input and output's current and past information, $x(k_i)$ is explicitly known. Therefore, an observer is not needed when using this state-space formulation. Also, note that the set-point information at k_i will be incorporated into the state variable $x(k_i)$ with its last element replaced by $y(k_i) - r(k_i)$.

Next, we proceed to the tutorial that will incorporate the analytical constraints into the optimal solution. In order to avoid confusion, the cases of constraints on plant input signal are presented first as these are the constraints most likely to be used in a practical application. The case of constraints on the output will follow in the second tutorial. Although the tutorials are presented for the DC motor control, they can be readily extended to an n th-order single-input and single-output system.

Tutorial 3.9. Write a MATLAB program for controlling a DC motor where both time constant and gain are normalized to yield $T = 1$ and $K = 1$. Sampling interval h is 0.1. The design parameters are $Q = C^T C$, and $R = 0.1$. The Laguerre parameters are tuning parameters, initially with $a = 0.5$ and $N = 1$. The prediction horizon $N_p = 46$. The tutorial will be divided into two parts. The first part deals with the constraints on input and second part deals with constraints on both input and output. The solutions are suitable for implementation on micro-controllers that have a limited computational power.

Part A. Constraints on u and Δu

Step by Step

1. Create a new file called *motor1.m*.
2. We will first discretize the continuous-time model, then make the state-space model. Enter the following program into the file:

```
K=1; % gain of the motor
T=1; % time constant of the motor
h=0.1; % sampling interval
num=K;
den=conv([1 0],[T 1]);
```

```
[numd,dend]=c2dm(num,den,h);
Ap=zeros(3,3);
Ap(1,:)=[-dend(1,2:3) numd(1,3)];
Ap(2,:)=[1 0 0];
Ap(3,:)=[0 0 0];
Bp(1,1)=numd(1,2);
Bp(2,1)=0;
Bp(3,1)=1;
Cp(1,1)=1;
Cp(1,2)=0;
Cp(1,3)=0;
Dp(1,1)=0;
```

3. Now, make the augmented state-space model. Enter the following program into the file:

```
n1=3;
m1=1;
n_in=1;
A=eye(n1+m1,n1+m1);
A(1:n1,1:n1)=Ap;
A(n1+1:n1+m1,1:n1)=Cp*Ap;
B=zeros(n1+m1,n_in);
B(1:n1,:)=Bp;
B(n1+1:n1+m1,:)=Cp*Bp;
C=zeros(m1,n1+m1);
C(:,n1+1:n1+m1)=eye(m1,m1);
```

4. Enter the design parameters and performance parameters and call the function DMPC.m to generate matrices required for the predictive control cost function J. Continue entering the following program into the file:

```
Q=C'*C;
R=0.3;
a=0.7;
N=1; %you can increase N
Np=46;
[Omega,Psi]=dmpe(A,B,a,N,Np,Q,R);
```

5. We can evaluate the closed-loop performance by calculating the feedback gain matrix and closed-loop system matrix. Continue entering the following program into the file:

```
[A1,L0]=lagd(a,N);
K_mpc=L0'*(Omega\Psi);
Acl=A-B*K_mpc;
Pole_close=eig(Acl);
```

6. Set up the initial conditions for closed-loop simulation using constrained control. We also specify the limits for Δu and u . Continue entering the following program into the file:

```

xm=zeros(n1,1);
N_sim=170;
u=0;
y=0;
y_delta_k=0;
y_delta_k_m1=0;
u_delta_k_m1=0;
r=[ones(1,200) ones(1,100) ones(1,200)];
Xf=[y_delta_k; y_delta_k_m1; u_delta_k_m1;
y-r(1,1)];
u_min=-0.3;
u_max=0.2;
deltau_min=-0.1;
deltau_max=0.1;
up=0.0;

```

7. The closed-loop simulation is performed recursively, and the optimal control is also found recursively. The analytical solution of optimal control is as discussed before.

```

for kk=1:N_sim;
eta=-(Omega\Psi)*Xf;
deltau=L0'*eta;
if (deltau>deltau_max) deltau=deltau_max;end
if (deltau<deltau_min)
deltau=deltau_min;end
u=up+deltau;
if (u>u_max) deltau=u_max-up;
u=u_max;
end
if (u<u_min) deltau=u_min-up;
u=u_min; end
deltau1(1,kk)=deltau; %save data
u1(1,kk)=u; %save data
y1(1,kk)=y; %save data
%%%%%
%plant simulation
%%%%%
yp=y;
xm_old=xm;
xm=Ap*xm+Bp*u; % calculate xm(k+1)
y=Cp*xm; %calculate y(k+1)
%updating feedback state variable Xf

```

```

y_delta_k_m1=y_delta_k;
y_delta_k=y-yp;
u_delta_k_m1=deltau;
Xf=[y_delta_k;
y_delta_k_m1;u_delta_k_m1;y-r(1,kk+1)];
up=u;
end

```

8. In a real-time implementation of the predictive control system on a DC motor, the four lines regarding plant simulation will not exist. Instead, the control signal u will be sent to the change on voltage and y will be taken as the measurement of the angular position. If the implementation uses a micro-controller, then storage of the data may not be permitted.
9. The simulation results are presented graphically. Continue entering the following program into the file:

```

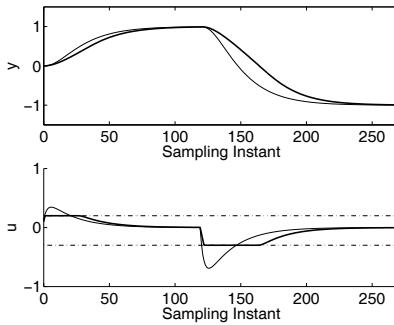
k=0:(N_sim-1);
figure(1)
subplot(211)
plot(k,y1,'b')
ylabel('y')
xlabel('Sampling Instant')
subplot(212)
plot(k,u1,'b')
ylabel('u')
xlabel('Sampling Instant')
figure(2)
plot(k,deltau1,'g')
ylabel('Delta u')
xlabel('Sampling Instant')

```

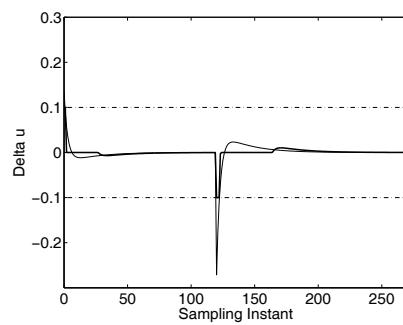
10. Run this program, and compare the control results with and without constraints. Figure 3.14 shows the comparison results. Indeed, both control u and difference of control Δu satisfy the constraints when they are imposed. It can be verified that this closed-form solution produces identical results when using quadratic programming.

Part B. Constraints on Output

The closed-form solution for constraints on output is slightly more complicated than the cases with constraints on input. The constraints on output, relatively speaking, need to be introduced cautiously because they could cause instability of the closed-loop system and severe deterioration of the closed-loop performance. The program below will show how to incorporate the output constraints in a closed-form solution. In the demonstration, we will assume



(a) Control signal and output



(b) Incremental control signal

Fig. 3.14. Comparison results of predictive control with and without constraints. Key: darker-solid-line, control with constraints; solid-line, control without constraints

that the set-point signal is zero, and the predictive control is to reject an input step disturbance. The program is built on the top of Part A of this tutorial.

Step by Step

1. Copy program `motor1.m` to `motor2.m`.
2. Before the closed-loop simulation, modify the constraints and add the lines to `motor2.m` that specify the simulation condition, the disturbance and constraints, including input and output.

```
r=[ones(1,120) -zeros(1,120) ones(1,200)]*0;
d=[ones(1,120) -zeros(1,120) ones(1,200)];
Xf=[y_delta_k; y_delta_k_m1;u_delta_k_m1;y-r(1,1)];
u_min=-1.5;
u_max=1.5;
deltau_min=-0.4;
deltau_max=0.4;
y_min=-0.14;
y_max=0.14;
M_act=C*B*L0';
E=C*A;
y_bar_min=y_min-r(1,1);
y_bar_max=y_max-r(1,1);
```

3. Change the simulation part of the program in `motor1.m` to the following paragraph, where the closed-form solution for output constraints is presented.

```
for kk=1:N_sim;
eta=-(Omega\Psi)*Xf;
```

```

deltau=L0'*eta;
beta=E*Xf+M_act*eta;
if (beta<y_bar_min)
    M_act1=-M_act;
lambda=-inv(M_act1*inv(Omega)*M_act1')*
(-y_bar_min+E*Xf+M_act1*(-eta));
eta=-inv(Omega)*(Psi*Xf+M_act1'*lambda);
deltau=L0'*eta;
end
if (beta>y_bar_max)
lambda=-inv(M_act*inv(Omega)*M_act')*
(y_bar_max-E*Xf+M_act*(-eta));
eta=-inv(Omega)*(Psi*Xf+M_act'*lambda);
deltau=L0'*eta; end
if(deltau>deltau_max) deltau=deltau_max;end
if (deltau<deltau_min)
deltau=deltau_min;end
u=u+deltau;
if (u>u_max) deltau=u_max-up; u=u_max;end
if (u<u_min) deltau=u_min-up; u=u_min; end
deltau1(1, kk)=deltau;
u1(1, kk)=u;
y1(1, kk)=y;
%%%%%
%plant simulation
%%%%%
yp=y;
xm_old=xm;
xm=Ap*xm+Bp*(u+d(kk));      % calculate xm(k+1)
y=Cp*xm;                      %calculate y(k+1)
%updating feedback state variable Xf
y_delta_k_m1=y_delta_k;
y_delta_k=y-yp;
u_delta_k_m1=deltau;
Xf=[y_delta_k; y_delta_k_m1; u_delta_k_m1; y-r(1, kk+1)];
y_bar_min=y_min-r(1, kk+1);
y_bar_max=y_max-r(1, kk+1);
up=u;
end

```

4. Run this program and compare the control results with and without constraints on the output.
5. Figure 3.15 shows the comparison of the control results. It is seen from these plots that all constraints are satisfied. However, in order to bring the

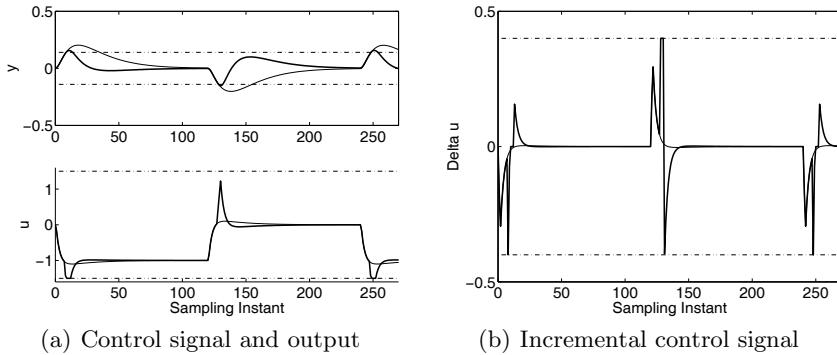


Fig. 3.15. Comparison results of predictive control with and without constraints. Key: darker-solid-line, control with constraints; solid-line, control without constraint; $y^{min} = -0.14$, $y^{max} = 0.14$

output within the constraints, the control signal u has a spike, also, Δu has a spike whenever the constraints on the output become active.

6. To demonstrate possible performance deterioration, choose smaller bounds for the output with $y^{min} = -0.12$ and $y^{max} = 0.12$, and simulate the closed-loop performance. Figure 3.16 shows that the closed-loop system responses oscillate within the constraints. This highlights the difficulties when imposing constraints on the plant output and that we need to take extra care if plant output constraints are necessary.

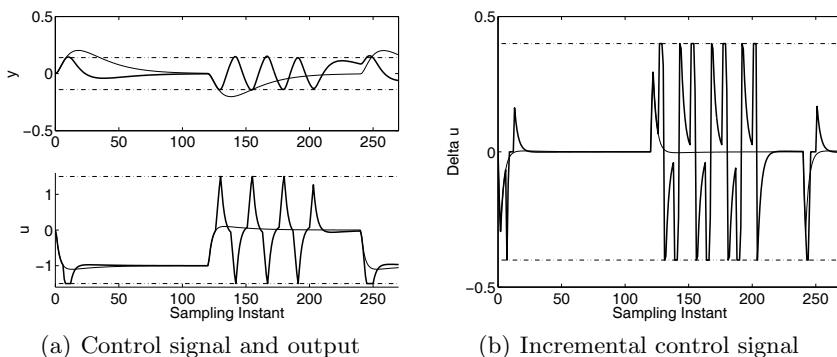


Fig. 3.16. Comparison results of predictive control with and without constraints. Key: darker-solid-line, control with constraints; solid-line, control without constraint; $y^{min} = -0.12$, $y^{max} = 0.12$

3.9 Summary

This chapter has proposed the design of discrete-time model predictive control using a set of Laguerre functions. The central idea is to express the future incremental control trajectory $\Delta u(k_i + m)$, $m = 0, 1, 2, \dots$, using an orthonormal expansion, and by doing so, the problem of finding the future incremental control trajectory is converted into one of finding the set of optimal coefficients for the expansion. A set of Laguerre functions, which is orthonormal, is used in the design. When the scaling factor $a = 0$, the predictive control designed using a set of Laguerre functions collapses to the original case of using the pulse operator where the coefficients of the expansion become identical to $\Delta u(k_i)$, $\Delta u(k_i + 1)$, \dots , $\Delta u(k_i + N)$.

In the literature of predictive control design, other analytical functions, such as exponential functions associated with the desired eigenvalues of the closed-loop system, have been used to capture the control trajectory (see Gawthrop and Ronco, 2002). Use of Laguerre functions is different from the other approaches. This is because the set of Laguerre functions form a group of candidate functions with appropriate orders, and with an increase of the number of terms, the orthonormal expansion will converge to the underlying optimal control trajectory. Thus, the accuracy is guaranteed to improve as the number of terms, N , increases. This could not be said if an arbitrary set of exponential functions or a set of other functions were put together without consideration of orthogonality.

The tuning of the predictive control system consists of the following steps:

1. Choose the weight matrix $Q = C^T C$ for minimization of the error between the set-point signal and output. This choice of Q produces excellent closed-loop performance when using the augmented state-space model, which virtually puts no weight on the incremental state variable Δx_m and identity weight matrix on the error between the set-point and output. R is selected as a diagonal matrix with smaller components corresponding to faster response speed.
2. Once Q and R are selected, the underlying optimal control trajectories are fixed. In general, a is selected as an estimate of the real part (absolute value) of the closed-loop dominant eigenvalue dictated by Q , R , and N is increased until the control trajectory no longer changes with the increase of N .
3. In some difficult applications, the closed-loop predictive control performance can be fine-tuned, input by input with the selection of the pair of parameters in the Laguerre functions, (a, N) . For instance, a larger a with a smaller number of N corresponds to a control signal with a slower decay rate. In any case, an increase of N will lead to the convergence of the control trajectory to the underlying trajectory dictated by the pair (Q, R) matrices.
4. The prediction horizon N_p is a tuning parameter, because the design model has an embedded integrator(s). Its value needs to be selected suffi-

ciently large to include the range within which the Laguerre functions decay to sufficiently small values. However, numerical ill-conditioning problems may occur, and they will be resolved in Chapter 4.

The idea is applicable to other orthonormal functions, such as a set of general orthonormal functions like Kautz functions (see Chapter 6). However, the Laguerre functions offer the advantage of simplicity for programming. In addition, the scaling factor and the number of terms are the tuning parameters without *a priori* knowledge of the desired closed-loop poles.

The initial idea of using Laguerre functions in the design of discrete-time model predictive control was published in Wang (2001a) and Wang (2004).

Problems

3.1. Discrete-time Laguerre models are used to describe stable dynamic systems. This class of models are called basis function models in the literature of system identification (see for example, Wang and Cluett, 2000, Wahlberg, 1991) in which a Laguerre model is estimated from plant experimental data. This type of model structure has been used in predictive control to describe the dynamics of a plant (see Finn *et. al.*, 1993). When using this class of models in the design of a predictive control system, an observer is not needed for the implementation of the predictive control system (see Problem 3.2). Suppose that a dynamic system is described by a continuous-time transfer function model $G(s)$, where $G(s)$ is

$$G(s) = \frac{e^{-3s}}{(10s + 1)^3}. \quad (3.75)$$

Approximate this continuous-time model by a discrete-time Laguerre transfer function model $G(z)$:

$$G_A(z) = \frac{\sqrt{1-a^2}}{1-az^{-1}} \left(c_1 + c_2 \frac{z^{-1}-a}{1-az^{-1}} + c_3 \left(\frac{z^{-1}-a}{1-az^{-1}} \right)^2 + \dots + c_n \left(\frac{z^{-1}-a}{1-az^{-1}} \right)^{n-1} \right). \quad (3.76)$$

A suggestion is to obtain the corresponding discrete-time transfer function model by discretization, based on which the discrete-time impulse response is generated. By following Tutorial 3.1, the coefficients of the discrete-time Laguerre model are obtained. The sampling interval Δt is 1, and $n = 3$ while the scaling factor a is 0.9.

3.2. Continue from Problem 3.1. Based on the description of the Laguerre function's state-space realization (3.5), choosing the state variable vector $x_m(k)$ corresponding to $L(k)$, A_m to A_l , B_m to $L(0)$, verify that by letting C_m be the Laguerre coefficient vector (*i.e.*, $C_m = [c_1 \ c_2 \ \dots \ c_n]$) the Laguerre

transfer function (3.76) is realized by this state-space model. With this state-space realization, the state variables are measurable through the input signal, thus an observer is not required. Find the augmented state-space model that has embedded an integrator in the Laguerre transfer function (3.76).

3.3. Continue from Problem 3.2. Based on the discrete-time state space model obtained from Problem 3.2 for the plant (3.75), design a discrete-time model predictive control system that will reject step input disturbance and follow a step set-point signal. Here, the design specifications are $Q = C^T C$, $R = 1$, $N_p = 30$, $N = 3$, and $a = 0.5$. Assuming that a step set-point signal $r = 1$ enters the system at $k = 0$ and a step input disturbance enters the system at $k = 30$, simulate the nominal closed-loop system response for $k = 0, 1, \dots, 100$ (use the Laguerre model $G_A(z)$ as the plant in the nominal closed-loop system simulation). The initial conditions of the state variables are assumed to be zero.

Repeat the simulation using the discrete-time plant model $G(z)$ obtained directly from $G(s)$ (3.75) and compare the closed-loop responses with the nominal closed-loop responses. What is the cause of the performance difference?

3.4. Laguerre models by nature are approximate models. In control system design, the higher the performance requirement, the more accurate the model needs to be. Improving accuracy of the approximation relies on an increase of the model term n in the Laguerre model. Repeat the Problems 3.1- 3.3 by using a different set of parameters. Firstly, increase the model order n in Problem 3.1 to 8, and observe the improvement of the Laguerre model. Secondly, maintaining all other parameters unchanged, use this higher-order Laguerre model in the design by repeating Problem 3.3. Compare the closed-loop responses with the results obtained from Problem 3.3.

3.5. An unstable, non-minimum phase system has the z-transfer function

$$G(z) = \frac{0.3(z - 1.1)}{(z - 0.3)(z - 1.5)},$$

where one of the poles as well as the zero are outside the unit circle of the complex plane. Design a discrete-time predictive control system with integral action. The weight matrices are $Q = C^T C$ and $R = 0.1$. Choose the prediction horizon as $N_p = 28$, and select $N = 3$, the scaling factor in the Laguerre function as $a = \frac{1}{1.5}$, which is the inverse of the unstable pole. Verify that this set of design parameters will produce a stable closed-loop predictive control system. The choice of prediction horizon N_p is particularly important for this unstable system (see Chapter 4). Therefore, reducing and increasing the prediction horizon N_p , observe the effects of this parameter on the closed-loop stability and numerical condition of the Hessian matrix Ω .

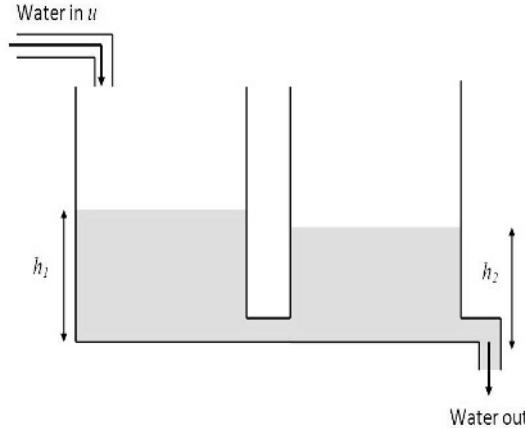


Fig. 3.17. Schematic plot for a double-tank process

3.6. A dynamic model of a vessel with two tanks connected in series is described by

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -\frac{\beta}{\alpha_1} & \frac{\beta}{\alpha_1} \\ \frac{\beta}{\alpha_2} & -\frac{\beta}{\alpha_2} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{\alpha_1} & 0 \\ 0 & \frac{1}{\alpha_2} \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad (3.77)$$

$$y(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}, \quad (3.78)$$

where $\alpha_1 > 0$ and $\alpha_2 > 0$ are the coefficients reflecting the dynamics of the first tank and second tank; β is the coefficient reflecting the connection between the two tanks; x_1 and x_2 are the liquid levels of the tanks; $u_1(t)$ and $u_2(t)$ are the control valves for the liquid flow into the tanks (see Figure 3.17). The coefficients are normalized and dimensionless. The double-tank has an integrating mode, which is revealed by examining the eigenvalues of the state-space model $(0, -1.5)$.

We assume that $\alpha_1 = 2$, $\alpha_2 = 1$ and $\beta = 1$; $\Delta t = 0.1$. Design a discrete-time MPC for this double tank so that both tank levels will follow step input references without steady-state errors. The weight matrices $Q = C^T C$ and $R = 0.1I$, where I is the identity matrix. The prediction horizon is selected as $N_p = 20$. Fixing $N_1 = N_2 = 3$, change $a_1 = a_2$ from 0.0, to 0.5 and to 0.8, observe the changes in the closed-loop step responses. If you do not want any over-shoot in the closed-loop responses, which Laguerre scaling parameters would you use?

3.7. Continue from Problem 3.6. We incorporate operational constraints into the predictive control system. With the design parameters as $a_1 = a_2 = 0.8$, $N_1 = N_2 = 3$, $N_p = 20$, $Q = C^T C$, and $R = 0.1I$, we compute the matrices Ω and Ψ in the cost function

$$J = \eta^T \Omega \eta + 2\Psi x(k_i).$$

With zero initial conditions, we fill the tanks where the step reference signals are $r_1 = 1$ and $r_2 = 0.5$ unit. After the levels reach the desired values, we then maintain the same reference signal for the second tank, whilst doubling the desired reference value for the first tank. To realize this, we could generate the references using the MATLAB command:

```
Nsim=100;
sp=0.5*ones(2,Nsim);
sp(1,1:N_sim/2)=ones(1,Nsim/2);
sp(1,N_sim/2+1:N_sim)=2*ones(1,Nsim/2);
```

where Nsim is the number of simulation samples. The constraints for the control signals are

$$-0.3 \leq \Delta u_1(k) \leq 0.3; -0.1 \leq \Delta u_2(k) \leq 0.1; 0 \leq u_1(k) \leq 2; -2 \leq u_2(k) \leq 0.5.$$

Simulate the predictive control system with constraints.

3.8. Extrusion is a continuous process in which a rotating screw is used to force the food material through the barrel of the machine and out through a narrow die opening. In this process the material is simultaneously transported, mixed, shaped, stretched and sheared under elevated temperature and pressure. Suppose that u_1 , u_2 , y_1 and y_2 represent screw speed, liquid pump speed, specific mechanical energy (SME) and motor torque, respectively. A continuous-time model for the food extruder is

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad (3.79)$$

where

$$\begin{aligned} G_{11} &= \frac{0.21048s + 0.00245}{s^3 + 0.302902s^2 + 0.066775s + 0.002186} \\ G_{12} &= \frac{-0.001313s^2 + 0.000548s - 0.000052}{s^4 + 0.210391s^3 + 0.105228s^2 + 0.00777s + 0.000854} \\ G_{21} &= \frac{0.000976s - 0.000226}{s^3 + 0.422036s^2 + 0.091833s + 0.003434} \\ G_{22} &= \frac{-0.00017}{s^2 + 0.060324s + 0.006836}. \end{aligned}$$

Design a discrete-time model predictive control system for this food extruder.

- Choosing sampling interval $\Delta t = 3$ sec, obtain the discrete-time food extruder model. Tutorial 3.3 shows how to convert individual transfer functions into an MIMO system and obtain a discrete state-space model.

2. Design a discrete model predictive controller that will track step set-point signals and reject constant disturbances. The design parameters are prediction horizon $N_p = 28$, $N_1 = N_2 = 4$, $a_1 = a_2 = 0.6$; and the weight matrices $R = I$ and $Q = C^T C$, where I is the identity matrix with an appropriate dimension. What is the closed-loop feedback control gain matrix and where are the closed-loop poles allocated in the complex plane?
3. Repeat the predictive control design with $R = 0$, and $R = 6I$. Compare the predictive control system parameters with those from case $R = I$. What are your observations?
4. Design an observer using MATLAB program ‘dlqr’

```
Kob= dlqr(A',C',Qob,Rob)'
```

where $Qob = I_1$, and $Rob = 0.1I_2$, I_1 and I_2 are the identity matrices with dimension equal to the number of states and to the number of outputs respectively. What is the observer gain vector Kob ?

5. Simulate the closed-loop predictive control system using an observer to estimate the state variables for reference following. The set-point signals are $r_1 = 3$ and $r_2 = 0$. The steady-state vector of the control signal is $uss = [300 \ 2000]^T$ and the steady-state vector of the output signal is $yss = [639 \ 42]^T$. Present your simulation results in terms of the actual control signals and output signals that contain their steady-state values (*i.e.*, $u_{act} = u + uss$, $y_{act} = y + yss$).
6. Simulate the closed-loop predictive control system using observer to estimate the state variables for disturbance rejection and noise attenuation. A white noise with standard deviation of 0.1 is added to output y_2 to simulate the situation of measurement noise in the system; and a unit step input disturbance enters the system at control signal u_2 after 40 sample periods into the simulation. Present your simulation results in terms of the actual control signals and output signals that contain their steady-state values.

- 3.9.** Continue from Problem 3.8. Incorporate operational constraints in the predictive control system. The design parameters are specified as $N_p = 28$, $N_1 = N_2 = 4$, $a_1 = a_2 = 0.8$; and the weight matrices $R = I$ and $Q = C^T C$, leading to the cost function:

$$J = \eta^T \Omega \eta + 2\eta^T \Psi \hat{x}(k_i).$$

The operational constraints are imposed on the changes of control signals as

$$-0.1 \leq \Delta u_1(k) \leq 0.1; \quad -0.1 \leq \Delta u_2(k) \leq 0.1,$$

which protect the screws and pumps from possible damage caused by large sudden demand changes. Simulate the predictive control system with an observer designed from Problem 3.8 by making a unit step set-point change to the motor torque, while maintaining SME as constant.

Discrete-time MPC with Prescribed Degree of Stability

4.1 Introduction

In this chapter, we discuss discrete-time model predictive control with a prescribed degree of stability. In Section 4.2, we begin with illustrative examples to show how the prediction horizon in the discrete-time model predictive control affects the stability and numerical condition of the algorithm. In Section 4.3, we propose the use of an exponentially weighted moving horizon window in model predictive control design, which converts the numerically ill-conditioned Hessian matrix into a numerically well-conditioned Hessian in the presence of a large prediction horizon. The solution is extended to predictive control with an infinite horizon. In Section 4.4, asymptotic stability for predictive control designed using an infinite horizon is achieved through exponential data weighting and modification of the weight matrices. In Section 4.5 we show how to design a predictive control system with a prescribed degree of stability. In Section 4.6, we discuss the tuning parameters in the design of model predictive control systems. In Section 4.7, we introduce exponentially weighted constrained control. In the final section (see Section 4.8), an additional benefit when using exponential data weighting is illustrated through a case study of predictive control of a complex system.

The results are established first by using the control vector Δu with the Laguerre pole $a = 0$, and then extended to the general case using Laguerre functions where $0 \leq a < 1$. The case of smaller N is of particular interest to us as an approximation to the optimal closed-loop performance (see Section 4.6).

4.2 Finite Prediction Horizon: Re-visited

In this section, we will investigate the sensitivity issues in the design of predictive control using finite prediction horizon.

Table 4.1. Eigenvalues and condition number κ of the Hessian matrix

N_p	$\lambda_{min}(\Omega)$	$\lambda_{max}(\Omega)$	$\kappa(\Omega)$
10	9.85	1.01×10^4	1.03×10^9
30	279.5	2.43×10^6	8.7×10^3
300	2.85×10^5	2.43×10^{11}	8.64×10^5

4.2.1 Motivational Example

Example 4.1. Suppose that the discrete-time model is of the form

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k), \end{aligned} \quad (4.1)$$

$$\text{where } A_m = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}; B_m = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}; C_m = [1 \ 0].$$

Illustrate the changes on the condition number of the Hessian matrix

$$\Omega = \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right),$$

with increase of the prediction horizon N_p , where the Laguerre pole $a = 0$, and $N = 2$. Assume $R_L = 0$ to simplify the problem.

Solution. To examine how the choice of prediction horizon affects the numerical condition of the Hessian matrix, we select the prediction horizon $N_p = 10, 30, 300$ respectively. The minimum and maximum eigenvalues of the Hessian matrix are calculated. The results are presented in Table 4.1. It is seen from Table 4.1 that the condition number of the Hessian matrix increases as the prediction horizon N_p increases.

4.2.2 Origin of the Numerical Conditioning Problem

The essence of model predictive control is to minimize the cost function J at a given time k_i by solving for the optimal control trajectory $\Delta u(\cdot)$ which is described by a set of coefficients and Laguerre functions, where J is defined as

$$J = \sum_{j=1}^{N_p} x(k_i + j | k_i)^T Q x(k_i + j | k_i) + \sum_{j=0}^{N_p} \Delta u(k_i + j)^T R \Delta u(k_i + j), \quad (4.2)$$

where $Q \geq 0$ and $R > 0$. The strength of MPC formulated using Laguerre functions lies in the fact that the cost function J is optimized in real time, subject to a set of linear inequality constraints:

$$M\eta \leq \gamma, \quad (4.3)$$

where $\Delta U = [\Delta u(k_i)^T \Delta u(k_i + 1)^T \dots \Delta u(k_i + N_p)^T]^T$. This cost function (4.2) is expressed in terms of the coefficients η and transformed into the cost, given by

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(k_i). \quad (4.4)$$

Recall that we embedded integrators in the state-space model (A, B, C) for obtaining integral action, and the prediction of the state variables (see (3.43)) involves the matrix power A^m and the convolution sum $\phi(m)$. The matrices Ω and Ψ are the following quantities:

$$\Omega = \sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L; \quad \Psi = \sum_{m=1}^{N_p} \phi(m) Q A^m.$$

When there is an integrator in the system matrix A , the norms of the matrix power $\|A^m\|$ and the convolution sum $\|\phi(m)\|$ do not decay to zero, as m increases. Thus, the magnitudes of the elements in Ω increase as the prediction horizon N_p increases. Hence, if the prediction horizon N_p is large, a numerical conditioning problem occurs. This problem exists in the majority of the classical predictive controllers formulations, including GPC and DMC. The traditional approaches to overcome this numerical problem include the use of an inner-loop state feedback stabilization of the design model (Kouvaritakis and Rossiter, 1992, Rossiter *et al.*, 1998) that may compromise the closed-loop performance when constraints become active, or the use of prediction horizon N_p and control horizon N_c as the tuning parameters (see for example Clarke *et al.*, 1987), which is exactly the technique that we used in the previous chapters. This finite horizon idea relies on the fact that the solution for η is based on

$$\eta = - \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right)^{-1} \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right) x(k_i), \quad (4.5)$$

where for a large prediction horizon N_p , we have the scenario of a large number divided by another large number. With a matrix inversion algorithm that improves the numerical accuracy of this type of situations, not surprisingly, finite horizon predictive control systems are used in many applications. This numerical problem becomes severe when the plant model itself is unstable, or when the dimension of the matrix A is large.

What we propose in the following is to modify the original cost function (4.2) so that it produces numerically well conditioned solutions in real time. By doing so, a long prediction horizon N_p can be used safely without jeopardizing the numerical stability of the algorithms. The target of our approach is to produce a predictive control system with a prescribed degree of closed-loop stability.

The development consists of three stages. In stage one, we will propose an intuitive approach for improving the numerical condition of MPC algorithms, but without guaranteeing closed-loop stability. Asymptotic stability will arrive at stage two. Stage three will show how to use the results to create a prescribed degree of closed-loop stability for the predictive control algorithm.

4.3 Use of Exponential Data Weighting

We will introduce the idea of exponential data weighting in the design of model predictive control systems. Use of exponential data weighting originated from Anderson and Moore (1971), where continuous-time systems were encountered in the LQR design and the exponential factor $e^{\lambda t}$ was used. The discrete counterpart to $e^{\lambda t}$ for $t \geq 0$ is the geometric sequence $\{\alpha^j, j = 0, 1, 2, \dots\}$, in which we set $\alpha = e^{\lambda \Delta t}$ with Δt being the sampling interval. Because of this link to the continuous-time counterpart, we call the discrete weights an exponential weighting set.

4.3.1 The Cost Function

For simplicity, we will first consider the case of $a = 0$ that corresponds to the case of shift operator, then extend the case to the more general case when $0 \leq a < 1$. Let us propose the cost function in the following form:

$$\begin{aligned} J = & \sum_{j=1}^{N_p} \alpha^{-2j} x(k_i + j | k_i)^T Q x(k_i + j | k_i) \\ & + \sum_{j=0}^{N_p} \alpha^{-2j} \Delta u(k_i + j)^T R \Delta u(k_i + j). \end{aligned} \quad (4.6)$$

If $\alpha = 1$, then the cost function becomes identical to the traditional cost function that has been used in the previous chapters. We are more interested in the cases when $\alpha \neq 0$.

Exponentially Increasing Weight

If $\alpha < 1$, the exponential weights $\alpha^{-2j}, j = 1, 2, \dots, N_p$, de-emphasizes the state $x(k_i + j | k_i)$ at the current time and places emphasis on those at the future time, because the weights increase as the sample index j increases. The continuous-time counterpart of this exponentially increasing weight was used in Anderson and Moore (1971) for LQR design. Their solutions were obtained through the use of a steady-state Riccati equation. The idea was used in receding horizon control (see Kwon and Han, 2005, Yoon and Clarke, 1993). The key in all the previous approaches relies on the use of an exponentially increasing weight in the cost function, along with the solution obtained analytically from a Riccati approach.

Exponentially Decreasing Weight

If $\alpha > 1$, the exponential weights $\alpha^{-2j}, j = 1, 2, \dots, N_p$ place more emphasis on the state $x(k_i + j | k_i)$ at the current time and less emphasis on those at

future times, because the weights decrease as the sample j increases. This is the case that will be used in the design of predictive control systems.

We will show the equivalence of the cost and progress to understanding the reasons why the exponentially decreasing weight is useful in the context of predictive control design.

4.3.2 Optimization of Exponentially Weighted Cost Function

The design of discrete-time model predictive control with exponential data weighting is to minimize the cost function J for a given $\alpha > 1$ where

$$\begin{aligned} J = & \sum_{j=1}^{N_p} \alpha^{-2j} x(k_i + j | k_i)^T Q x(k_i + j | k_i) \\ & + \sum_{j=0}^{N_p} \alpha^{-2j} \Delta u(k_i + j)^T R \Delta u(k_i + j), \end{aligned} \quad (4.7)$$

subject to inequality constraints expressed by

$$M \Delta U \leq \gamma, \quad (4.8)$$

and the state-equation constraint:

$$x(k_i + j + 1 | k_i) = Ax(k_i + j | k_i) + B \Delta u(k_i + j), \quad (4.9)$$

where M and γ are the data matrices for the constraints and the parameter vector

$$\Delta U^T = [\Delta u(k_i)^T \Delta u(k_i + 1)^T \Delta u(k_i + 2)^T \dots \Delta u(k_i + N_p)^T].$$

The exponentially weighted cost function by itself does not reveal much about its role in predictive control design. Its role is revealed through the transformed variables. Define the sequence of exponentially weighted incremental control:

$$\Delta \hat{U}^T = [\alpha^{-0} \Delta u(k_i)^T \alpha^{-1} \Delta u(k_i + 1)^T \dots \alpha^{-N_p} \Delta u(k_i + N_p)^T],$$

and the exponentially weighted state variable:

$$\hat{X}^T = [\alpha^{-1} x(k_i + 1 | k_i)^T \alpha^{-2} x(k_i + 2 | k_i)^T \dots \alpha^{-N_p} x(k_i + N_p | k_i)^T].$$

By using these exponentially weighted variables, the exponentially weighted cost function is expressed in terms of the transformed variables. The result is summarized by the theorem below.

Theorem 4.1. *The minimum solution of the exponentially weighted cost function J (4.7) subject to the inequality constraints (4.8) and state-equation constraints (4.9) can be found by minimizing*

$$\hat{J} = \sum_{j=1}^{N_p} \hat{x}(k_i + j \mid k_i)^T Q \hat{x}(k_i + j \mid k_i) + \sum_{j=0}^{N_p} \Delta \hat{u}(k_i + j)^T R \Delta \hat{u}(k_i + j), \quad (4.10)$$

subject to

$$M_\alpha \Delta \hat{U} \leq \gamma, \quad (4.11)$$

where $\hat{x}(k_i + j \mid k_i)$ and $\Delta \hat{u}(k_i + j)$ are governed by the following difference equation:

$$\hat{x}(k_i + j + 1 \mid k_i) = \frac{A}{\alpha} \hat{x}(k_i + j \mid k_i) + \frac{B}{\alpha} \Delta \hat{u}(k_i + j), \quad (4.12)$$

where M_α is the matrix defined by

$$M_\alpha = M \begin{bmatrix} I & 0 & \dots & 0 & 0 \\ 0 & \alpha^1 I & \dots & 0 & 0 \\ 0 & 0 & \dots & \alpha^{N_p-1} I & 0 \\ 0 & 0 & \dots & 0 & \alpha^{N_p} I \end{bmatrix}.$$

Proof. To find the optimal solution for the exponentially weighted cost and constraints by (4.7) and (4.9), we let

$$\hat{x}(k_i + j \mid k_i) = \alpha^{-j} x(k_i + j \mid k_i)$$

$$\Delta \hat{u}(k_i + j) = \alpha^{-j} \Delta u(k_i + j).$$

Then the exponentially weighted cost function J (4.7) equals the expression of \hat{J} given by (4.10). Note that the state variable model (4.9) can also be written in terms of $\hat{x}(k_i)$ and $\Delta \hat{u}(k_i)$. At a future sample j , multiplying (4.9) by the factor $\alpha^{-(j+1)}$ leads to

$$\begin{aligned} \hat{x}(k_i + j + 1 \mid k_i) &= \alpha^{-(j+1)} x(k_i + j + 1 \mid k_i) \\ &= \alpha^{-1} A \hat{x}(k_i + j \mid k_i) + \alpha^{-1} B \Delta \hat{u}(k_i + j), \end{aligned} \quad (4.13)$$

where we used the relationship that

$$\alpha^{-j} x(k_i + j \mid k_i) = \hat{x}(k_i + j \mid k_i); \quad \alpha^{-j} \Delta u(k_i + j) = \Delta \hat{u}(k_i + j).$$

In addition, the original inequality constraints can be transformed into the constraints (4.11) by substituting $\Delta u(k_i) = \Delta \hat{u}(k_i)$, $\Delta u(k_i + 1) = \alpha \Delta \hat{u}(k_i + 1)$, \dots , $\Delta u(k_i + j) = \alpha^j \Delta \hat{u}(k_i + j)$. Thus, with the identical initial condition at the present time k_i , i.e., $\hat{x}(k_i) = x(k_i)$ and $\Delta \hat{u}(k_i) = \Delta u(k_i)$, the cost function presented by (4.7) subject to the inequality constraints (4.8) and state variable constraints (4.9) is identical to the cost function (4.10) subject to the inequality constraints (4.11) and state variable constraints (4.9).

This result shows that if we choose a cost function with exponential data weight, the equivalent result is to minimize the unweighted cost function,

but with a re-scaling applied to the design model. As we discussed previously, instability of the design model is the cause of the numerical problem occurring in the class of predictive control systems that have embedded integrators. Therefore, with a suitable choice of scaling factor α , the original unstable design model could be substituted with the transformed design model that is stable with all poles inside the unit circle.

Example 4.2. Consider the same double-integrator system given in Example 4.1. Examine how the parameter α used in the weighting affects the numerical condition and closed-loop control performance with constraints on the amplitude of the control signal as

$$-12 \leq u(k) \leq 12.$$

Solution. We make a simple modification to the design of a model predictive controller by using the exponential weight factor α . We use the following MATLAB scripts to generate Ω and Ψ matrices.

```
%A_e and B_e are the matrices for
%the augmented state space model.
alpha=1.2;
[Omega,Psi]=dmpc(A_e/alpha,B_e/alpha,a,N,Np,Q,R);
```

We only impose constraints on the first sample of the control, therefore, there is no change at the constraint formulation.

The closed-loop responses for $\alpha = 1$ with the three different prediction horizons are shown in Figure 4.1a. It is seen that when the prediction horizon changes, the closed-loop responses differ. In contrast, it is shown in Figure 4.1b that with exponential data weighting $\alpha = 1.2$, the closed-loop responses become invariant after a certain prediction horizon.

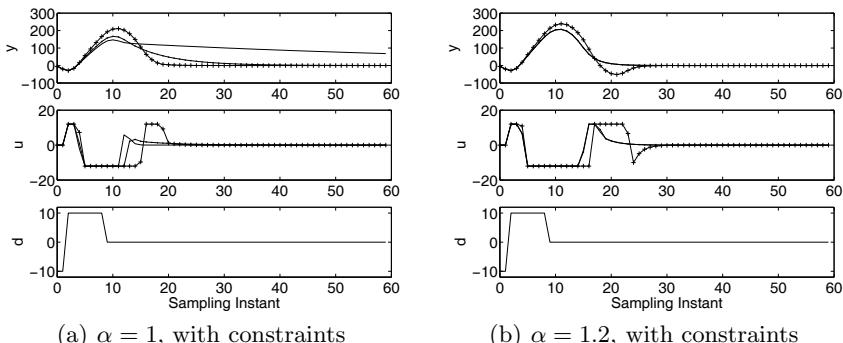


Fig. 4.1. Closed-loop responses using different prediction horizons. Top figure, output response; middle figure, control signal response; bottom figure, input disturbance. Key: solid line $N_p = 300$; solid-dotted line $N_p = 30$; solid-plus line: $N_p = 10$

Table 4.2. Eigenvalues and condition number of the Hessian matrix Ω

N_p	α	$\lambda_{min}(\Omega)$	$\lambda_{max}(\Omega)$	$\kappa(\Omega)$
10	1/1.2	69.5	2.44×10^5	3.52×10^9
30	1/1.2	3.46×10^5	4.59×10^{10}	1.32×10^5
300	1/1.2	3.1×10^{48}	3.5×10^{57}	1.13×10^9
10	1	9.85	1.01×10^4	1.03×10^3
30	1	279.5	2.43×10^6	8.7×10^3
300	1	2.85×10^5	2.43×10^{11}	8.64×10^3
10	1.2	1.98	561.4	283.5
30	1.2	5.0	1.83×10^3	365.2
300	1.2	5.11	1.86×10^3	362.9

To examine how the choice of prediction horizon affects the numerical condition of the Hessian matrix, we select the parameters in the Laguerre functions as $N = 2$ and $a = 0$, prediction horizon $N_p = 10, 30, 300$, respectively. Table 4.2 shows the changes in eigenvalues and the condition number of the Hessian matrix with respect to N_p . For comparison purpose, the results have been presented for the cases of $\alpha = 1/1.2$ (exponentially increasing weight), $\alpha = 1$ (no exponential weighting) and $\alpha = 1.2$ (exponentially decreasing weighting). From this table, it is seen that with exponentially increasing weighting, the Hessian matrix is poorly conditioned even for short prediction horizon; without exponential weighting the condition number increases rapidly as the prediction horizon increases. However, with exponentially decreasing data weighting, the condition number converges to a finite value and is much smaller than the one obtained without using exponential weighting. Obviously, it is not feasible to use exponentially increasing weighting in this context, as the numerical condition rapidly deteriorates as prediction horizon increases, when $\alpha < 1$.

4.3.3 Interpretation of Results from Exponential Weighting

The key point is that by transforming the exponentially weighted cost function to the traditional cost function, the augmented state-space model (A, B) is changed into $(\alpha^{-1}A, \alpha^{-1}B)$. Suppose that the $|\lambda_{max}(A)|$ denotes the maximum modulus of all the eigenvalues of A . Thus, by choosing $\alpha > |\lambda_{max}(A)|$, the system matrix $\alpha^{-1}A$ is transformed to a matrix with the maximum modulus of all eigenvalues being less than 1. For instance, assuming that the plant model is stable and the augmented system matrix A contains eigenvalues equal to unity, then by choosing $\alpha = 1.1$, all eigenvalues of A are divided by 1.1, and the transformed design model is asymptotically stable with all eigenvalues within the unit circle in the complex plane. Hence, the numerical problems associated with the class of predictive control systems are resolved, because of the stable design model.

Because the optimization is performed using the exponentially weighted variables, the Hessian matrix has a significantly reduced numerical condition number. Furthermore, with receding horizon control, upon finding the solution, the actual control $\Delta u(k_i) = \Delta \hat{u}(k_i)$.

With this simple modification, intuitively we understand that there is no guarantee on the closed-loop stability with an arbitrary choice of $\alpha > 1$. However, when α is chosen to be slightly larger than one for the class of stable plants with embedded integrator, the closed-loop predictive systems are often found to be stable with $Q = C^T C$ and a diagonal R matrix with small positive elements. In addition, closed-loop stability needs to be checked by examining the eigenvalues of the closed-loop system when constraints are not activated. To make sure that we understand the issues properly, they are investigated further.

With the exponentially weighted cost function, for the first time, the prediction horizon N_p can be selected to be sufficiently large to approximate the infinite prediction horizon case. Thus with $Q \geq 0$ and $R > 0$, and sufficiently large $N_p (\rightarrow \infty)$, minimizing

$$J = \sum_{j=1}^{N_p} \hat{x}(k_i + j | k_i)^T Q \hat{x}(k_i + j | k_i) + \sum_{j=0}^{N_p} \Delta \hat{u}(k_i + j)^T R \Delta \hat{u}(k_i + j)$$

$$\hat{x}(k+1) = \frac{A}{\alpha} \hat{x}(k) + \frac{B}{\alpha} \Delta \hat{u}(k)$$

is equivalent to the discrete-time linear quadratic regulator (DLQR) problem.

The traditional DLQR problem is solved using the algebraic Riccati equation shown in (4.14). The basic results are summarized below. For those who are interested in further study of this topic, Kailath (1980) gives a good introduction. For our particular problem, suppose that the pair $(\alpha^{-1}A, \alpha^{-1}B)$ is controllable, and the pair $(\alpha^{-1}A, C)$ is observable where $Q = C^T C$. Then there is a stabilizing state feedback control gain matrix K where

$$K = (R + \alpha^{-2} B^T P_\infty B)^{-1} \alpha^{-2} B^T P_\infty A$$

$$\frac{A^T}{\alpha} [P_\infty - P_\infty \frac{B}{\alpha} (R + \frac{B^T}{\alpha} P_\infty \frac{B}{\alpha})^{-1} \frac{B^T}{\alpha} P_\infty] \frac{A}{\alpha} + Q - P_\infty = 0, \quad (4.14)$$

so that the closed-loop system

$$\hat{x}(k_i + j + 1 | k_i) = \alpha^{-1} (A - BK) \hat{x}(k_i + j | k_i) \quad (4.15)$$

is stable with all its eigenvalues inside the unit circle. Thus for a given bounded initial state condition $\hat{x}(k_i)$, there exists an ϵ where $0 < \epsilon < 1$ such that the transformed state variable satisfies

$$\|\hat{x}(k_i + j | k_i)\| \leq (1 - \epsilon)^j \|\hat{x}(k_i)\|. \quad (4.16)$$

Because $x(k_i + j \mid k_i) = \alpha^j \hat{x}(k_i + j \mid k_i)$, then with the identical initial condition $x(k_i) = \hat{x}(k_i)$,

$$\|x(k_i + j \mid k_i)\| \leq (\alpha \times (1 - \epsilon))^j \|x(k_i)\|. \quad (4.17)$$

Both (4.15) and (4.17) point to the same conclusion. Namely, this choice of exponentially weighted cost function will produce a stable closed-loop system if

$$\alpha \times (1 - \epsilon) < 1.$$

We can also examine the closed-loop eigenvalues of the original system. From (4.15), the transformed system matrix $\alpha^{-1}(A - BK)$ has all its eigenvalues inside the unit circle by taking the prediction horizon N_p to infinity, which is

$$\alpha^{-1} |\lambda_{\max}(A - BK)| < 1,$$

therefore, the actual closed-loop system $(A - BK)$ has eigenvalues

$$|\lambda_{\max}(A - BK)| < \alpha.$$

By choosing $\alpha > 1$, there is no guarantee that the closed-loop of the original system will be stable. But, if α is chosen to be slightly larger than unity, then the closed-loop system $A - BK$ would often be stable. Indeed, a large number of simulation tests show that this simple modification usually produces a stable closed-loop system, if the unstable modes from the augmented model come from the embedded integrators. However, a proper choice of the weight matrices Q and R is important to create the degree of stability $1 - \epsilon$ for the transformed system.

4.4 Asymptotic Closed-loop Stability with Exponential Weighting

Further work is required to produce a predictive control system with guaranteed stability as well as a numerically well-conditioned solution.

4.4.1 Modification of Q and R Matrices

In order to produce an asymptotic closed-loop predictive control system with exponential data weighting, the matrices Q and R need to be selected to produce the closed-loop system with stability margin $\frac{1}{\alpha}$. Better still is the selection of a new Q and a new R to achieve the same original cost function when the exponential data weight was not used.

The basic idea is as follows. The exponentially decreasing weight $\alpha > 1$ increased the magnitudes of the actual closed-loop eigenvalues by the α factor. If the new Q and R matrices are selected to decrease the magnitudes of the

eigenvalues of the exponentially weighted system by a factor of α^{-1} , then the magnitudes of the actual closed-loop eigenvalues become unchanged from this exercise. This basic idea is summarized by the theorem below, which establishes the relationship between the original cost and the weighted cost function by selecting the new Q and R matrices.

Theorem 4.2. *Subject to the same system state equation*

$$x(k_i + j + 1 | k_i) = Ax(k_i + j | k_i) + B\Delta u(k_i + j), \quad (4.18)$$

the optimal solution of $\Delta u(k_i + j)$ by minimizing the cost function J_α defined as

$$\begin{aligned} J_\alpha &= \sum_{j=1}^{\infty} \alpha^{-2j} x(k_i + j | k_i)^T Q_\alpha x(k_i + j | k_i) \\ &\quad + \sum_{j=0}^{\infty} \alpha^{-2j} \Delta u(k_i + j)^T R_\alpha \Delta u(k_i + j) \end{aligned} \quad (4.19)$$

is identical to the solution found by minimizing the original cost

$$J = \sum_{j=1}^{\infty} x(k_i + j | k_i)^T Q x(k_i + j | k_i) + \sum_{j=0}^{\infty} \Delta u(k_i + j)^T R \Delta u(k_i + j), \quad (4.20)$$

where the new Q_α and R_α are selected according to

$$\gamma = \frac{1}{\alpha} \quad (4.21)$$

$$Q_\alpha = \gamma^2 Q + (1 - \gamma^2) P_\infty \quad (4.22)$$

$$R_\alpha = \gamma^2 R, \quad (4.23)$$

and P_∞ is the solution of the algebraic Riccati equation as below:

$$A^T [P_\infty - P_\infty B(R + B^T P_\infty B)^{-1} B^T P_\infty] A + Q - P_\infty = 0. \quad (4.24)$$

Proof. For a fixed initial condition $x(k_i)$, the optimal solution to the cost function (4.20), when constraints are not activated, is given by the Riccati equation

$$A^T (P_\infty - P_\infty B(R + B^T P_\infty B)^{-1} B^T P_\infty) A + Q - P_\infty = 0 \quad (4.25)$$

$$K = - (R + B^T P_\infty B)^{-1} B^T P_\infty A. \quad (4.26)$$

From Section 4.3.3, that the optimal solution of the exponentially weighted cost function (4.19) is given by the Riccati equation

$$\bar{K} = (R_\alpha + \alpha^{-2} B^T \bar{P}_\infty B)^{-1} \alpha^{-2} B^T \bar{P}_\infty A \quad (4.27)$$

$$\frac{A^T}{\alpha}[\bar{P}_\infty - \bar{P}_\infty \frac{B}{\alpha}(R_\alpha + \frac{B^T}{\alpha}\bar{P}_\infty \frac{B}{\alpha})^{-1}\frac{B^T}{\alpha}\bar{P}_\infty]\frac{A}{\alpha} + Q_\alpha - \bar{P}_\infty = 0. \quad (4.28)$$

We will next verify that (4.25) is equal to (4.28) and (4.26) is equal to (4.27).

From the Riccati equation (4.25), we multiply each A and B with $\frac{\alpha}{\gamma}$ and then let $\hat{A} = \frac{A}{\alpha}$ and $\hat{B} = \frac{B}{\alpha}$. The Riccati equation (4.25) becomes

$$\frac{\hat{A}^T}{\gamma}[P_\infty - P_\infty \frac{\hat{B}}{\gamma}(R + \frac{\hat{B}^T}{\gamma}P_\infty \frac{\hat{B}}{\gamma})^{-1}\frac{\hat{B}^T}{\gamma}P_\infty]\frac{\hat{A}}{\gamma} + Q - P_\infty = 0, \quad (4.29)$$

where $\gamma = \frac{1}{\alpha}$. Multiplying γ^2 by both sides of (4.29) leads to

$$\hat{A}^T[P_\infty - P_\infty \hat{B}(R\gamma^2 + \hat{B}^T P_\infty \hat{B})^{-1}\hat{B}^T P_\infty]\hat{A} + \gamma^2 Q - \gamma^2 P_\infty = 0. \quad (4.30)$$

Adding and subtracting P_∞ from (4.30) and letting

$$Q_\alpha = \gamma^2 Q - \gamma^2 P_\infty + P_\infty; \quad R_\alpha = \gamma^2 R,$$

the Riccati equation (4.30) becomes identical to (4.28). Hence, $\bar{P}_\infty = P_\infty$. Applying the same procedure to K given by (4.26) leads to \bar{K} . Hence $\bar{K} = K$. This establishes that the optimal solution of the exponentially weighted cost function J_α by (4.19) is identical to the one from the commonly used cost function J (4.20).

4.4.2 Interpretation of the Results

The essence of the results lies in the fact that the two cost functions lead to the same optimal control. However, the commonly used cost function is limited to a finite prediction horizon for the class of predictive control algorithms that have embedded integrators. In contrast, the exponentially weighted cost function removes the problem because the model used in the prediction is modified to be stable using the factor α . As a result, the prediction horizon N_p can be selected to be sufficiently large without numerical problems. Hence, asymptotic closed-loop stability is guaranteed.

Example 4.3. Consider the simple double-integrator system described in 4.1 where $A_m = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}; B_m = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}; C_m = [1 \ 0]$.

Design a predictive control system with an integrator for disturbance rejection, where $Q = C^T C$, $R = 1$, $N_p = 150$, $N = 4$, $a = 0$. Calculate the closed-loop eigenvalues, gain matrix via the cost function using exponential data weighting with $\alpha = 1.6$ and compare the results with the case without weighting ($\alpha = 1$).

Solution.

With exponential data weighting

For the given Q and R , we solve the steady-state Riccati equation

$$A^T (P_\infty - P_\infty B(R + B^T P_\infty B)^{-1} B^T P_\infty) A + Q - P_\infty = 0$$

to obtain

$$P_\infty = \begin{bmatrix} 1.7367 & 1.6663 & 1.2633 \\ 1.6663 & 2.0965 & 1.0000 \\ 1.2633 & 1.0000 & 2.1663 \end{bmatrix}.$$

With this P_∞ , Q_α and R_α are calculated as

$$Q_\alpha = 0.625^2 Q + (1 - 0.625^2) P_\infty = \begin{bmatrix} 1.0583 & 1.0154 & 0.7698 \\ 1.0154 & 1.2775 & 0.6094 \\ 0.7698 & 0.6094 & 1.7107 \end{bmatrix}$$

$$R_\alpha = 0.625^2 R = 0.3906.$$

With this set of design parameters, the cost function used in the receding horizon control is calculated for a given $x(k_i)$

$$\begin{aligned} J &= \eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q_\alpha \phi(m)^T + R_\alpha I \right) \eta \\ &\quad + 2\eta^T \left(\sum_{m=1}^{N_p} \phi(m) Q_\alpha \hat{A}^m \right) x(k_i), \end{aligned} \quad (4.31)$$

where the pair $(\alpha^{-1}A, \alpha^{-1}B)$ is used in the computation to replace the pair (A, B) in the predictive control algorithm. The results are

$$\begin{aligned} \sum_{m=1}^{N_p} \phi(m) Q_\alpha \phi(m)^T + R_\alpha I &= \begin{bmatrix} 24.7384 & 23.0396 & 20.4633 & 17.3134 \\ 23.0396 & 24.7384 & 23.0396 & 20.4633 \\ 20.4633 & 23.0396 & 24.7384 & 23.0396 \\ 17.3134 & 20.4633 & 23.0396 & 24.7384 \end{bmatrix} \\ \sum_{m=1}^{N_p} \phi(m) Q_\alpha \hat{A}^m &= \begin{bmatrix} 11.0121 & 29.8538 & 3.0071 \\ 8.7619 & 27.4206 & 1.8794 \\ 6.6508 & 23.7887 & 1.1746 \\ 4.8909 & 19.7589 & 0.73 \end{bmatrix}. \end{aligned}$$

The condition number of the Hessian matrix is 124.6. The state feedback control gain matrix is the first row of the matrix

$$\begin{aligned} &\left(\sum_{m=1}^{N_p} \phi(m) Q_\alpha \phi(m)^T + R_\alpha I \right)^{-1} \left(\sum_{m=1}^{N_p} \phi(m) Q_\alpha \hat{A}^m \right) \\ &= \begin{bmatrix} 0.7978 & 1.2634 & 0.3678 \\ -0.1952 & 0.0594 & -0.1992 \\ -0.1793 & -0.1010 & -0.0950 \\ -0.0321 & -0.0405 & 0.0255 \end{bmatrix}. \end{aligned} \quad (4.32)$$

From the data shown so far, the predictive control system numerically behaves well. To confirm the result that this is equivalent to the DLQR solution, the MATLAB function `dlqr` with the Q and R parameter matrices is used to obtain the gain vector as

$$K_{dlqr} = [0.7980 \ 1.2633 \ 0.3684].$$

In comparison, there is a small discrepancy between the gain vectors calculated from DMPC and `dlqr`. However, this discrepancy can be further reduced by increasing the number of terms in the Laguerre functions, for example, from $N = 4$ to $N = 6$. By doing so the quantity in (4.32) becomes

$$\begin{bmatrix} 0.7980 & 1.2634 & 0.3684 \\ -0.1919 & 0.0606 & -0.1949 \\ -0.1757 & -0.0992 & -0.0913 \\ -0.0515 & -0.0457 & -0.0037 \\ 0.0017 & -0.0055 & 0.0145 \\ 0.0139 & 0.0092 & 0.0101 \end{bmatrix}.$$

By inspection of the first row, it is seen that the predictive control gain vector is identical to the gain vector from `dlqr` function (at least up to three digits).

Without exponential data weighting ($\alpha = 1$)

When $\alpha = 1$, the pair (A, B) is used in the computation of the predictive control system. The data matrices are

$$\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + RI = 10^8 \begin{bmatrix} 3.8604 & 3.7966 & 3.7333 & 3.6706 \\ 3.7966 & 3.7339 & 3.6717 & 3.6101 \\ 3.7333 & 3.6717 & 3.6107 & 3.5501 \\ 3.6706 & 3.6101 & 3.5501 & 3.4907 \end{bmatrix}$$

$$\sum_{m=1}^{N_p} \phi(m) Q A^m = 10^8 \begin{bmatrix} 0.0641 & 3.8925 & 0.0006 \\ 0.0630 & 3.8281 & 0.0006 \\ 0.0619 & 3.7643 & 0.0005 \\ 0.0608 & 3.7010 & 0.0005 \end{bmatrix}.$$

The condition number of the Hessian matrix is 1.4347×10^9 . The quantity in (4.32) is

$$\begin{bmatrix} 1.0338 & 1.4553 & 0.5068 \\ -0.6706 & -0.1772 & -0.5570 \\ -0.7583 & -0.5100 & -0.4055 \\ 0.3951 & 0.2318 & 0.4556 \end{bmatrix}.$$

Comparing the first row of this quantity with the `dlqr` gain matrix, it is seen that there are much larger errors for each component in this gain matrix, due to the numerical instability of the data matrices.

The next example shows the use of exponential data weighting in the predictive control design of a MIMO system.

Example 4.4. Consider the continuous-time system with transfer function representation

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{12.8(-s+4)^2}{(16.7s+1)(s+4)^2} & \frac{-1.89(-3s+4)^2}{(21.0s+1)(3s+4)^2} \\ \frac{1.28(-7s+4)^2}{(10.9s+1)(7s+4)^2} & \frac{-19.4(-3s+4)^2}{(14.4s+1)(3s+4)^2} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}.$$

The system is sampled using the interval $\Delta t = 1$. Integrators are used in the design for disturbance rejection. Assuming that $Q = C^T C$, and $R = I$; $a = 0$, $N = 5$ and $N_p = 140$, compute the solution using the long prediction horizon with exponential data weight $\alpha = 1.2$, and modified Q_α and R_α . Compare the eigenvalues and the gain matrices with DLQR solution.

Solution. First, we compute the DLQR solution to obtain the Riccati matrix P . Then, the P is used to obtain matrices Q_α and R_α , where $\gamma = \frac{1}{\alpha} = 0.8333$

$$Q_\alpha = 0.8333^2 Q + (1 - 0.8333^2) P \quad (4.33)$$

$$R_\alpha = 0.8333^2 R. \quad (4.34)$$

The MATLAB® code is

```
[K,P,E]=dlqr(A,B,Q,R);
gamma=0.8333;
Q_alpha=gamma^2*Q+(1-gamma^2)*P;
R_alpha=gamma^2*R;
```

The new Q_α and R_α matrices are used in the design of discrete-time model predictive control with exponential data weighting. Due to the large dimension of the system, instead of using tables, we present the comparison results graphically. Figure 4.2 is the comparison of closed-loop eigenvalues between the DMPC and DLQR where the results are seen to be identical to each other; and this is also verified by the comparison of the control gain matrices (see Figure 4.3); the condition number of the Hessian matrix is 475. In contrast, without exponential weighting, the condition number of the Hessian matrix is 44607, and the numerical solution is ill-conditioned.

Example 4.5. Suppose that a continuous-time system is described by the transfer function

$$G(s) = \frac{1}{s^2 + 2\xi\omega s + \omega^2},$$

where $\xi = 0.5$ and $\omega = 3$. Suppose that the sampling interval $\Delta t = 0.1$, and the weight matrices are $Q = C^T C$ with $C = [0 \ 0 \ 1]$ and $R = 0.3$. The prediction horizon is chosen to be $N_p = 46$. The parameters for the Laguerre functions are $a = 0.7$ and $N = 18$. We assume that at sample index $k_i = 10$,

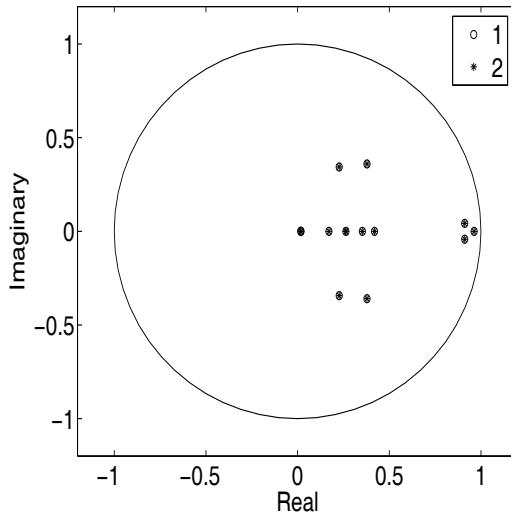


Fig. 4.2. Closed-loop eigenvalues ($\alpha = 1.2$). Key: (1) from DLQR; (2) from DMPC

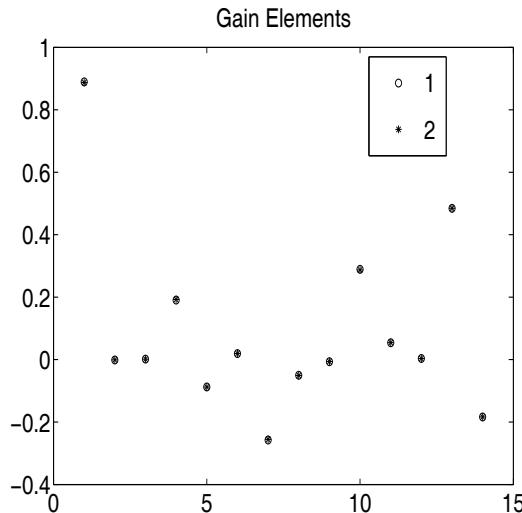


Fig. 4.3. Parameters in the first row of the control gain matrix ($\alpha = 1.2$). Key: (1) from DLQR; (2) from DMPC

the state variable $x(k_i) = [0.1 \ 0.2 \ 0.3]^T$. With $\alpha = 1.2$ and without constraints, illustrate the equivalence within one optimization window between LQR (scaled) and the exponentially weighted predictive control. In addition, show that when receding horizon control is applied, the closed-loop control systems are identical.

Solution. The feedback control gain vector K using DLQR program is

$$K = [1.7022 \ 5.6548 \ 1.6689],$$

which is identical to the predictive control gain vector generated using

$$K_{mpc} = L(0)^T(\Omega^{-1}\Psi) = [1.7022 \ 5.6548 \ 1.6689].$$

Within one optimization window, the optimal parameter vector for the Laguerre function is

$$\eta = -\Omega^{-1}\Psi x(k_i).$$

With this η , the transformed $\Delta\hat{u}(k_i + j)$ is constructed where

$$\Delta\hat{u}(k_i + j) = L(j)^T\eta.$$

Then, this $\Delta\hat{u}(k_i + j)$ is used to generate the closed-loop state trajectory $\hat{x}(k_i + j | k_i)$ based on the pair $(\alpha^{-1}A, \alpha^{-1}B)$.

To emphasize the transformation, the optimal LQR control trajectory is compared in the transformed variables, where by applying the controller K to the system specified by the pair $(\alpha^{-1}A, \alpha^{-1}B)$, the closed-loop control signal is $u(k_i + j) = -K(\alpha^{-1}A - \alpha^{-1}BK)^jx(k_i)$ and the closed-loop state variable in the transformed version is $x(k_i + j | k_i) = (\alpha^{-1}A - \alpha^{-1}BK)^jx(k_i)$. Numerical results show identical results between the transformed variables in DLQR system and the predictive control system within one optimization window. These identical plots are shown in Figure 4.4. Because predictive control uses the principle of receding horizon control, at $j = 0$ the first sample of the optimization window, the weight factor α^0 is unity. Thus, the unconstrained control should be completely identical to the optimal LQR solution when receding horizon control is applied. In the closed-loop simulation, a step input signal is applied at sampling time $k = 0$, and the predictive control systems are compared for the cases. Numerical results show that the closed-loop responses are identical. These identical responses are plotted in Figure 4.5. This is very good because the scaling factor is not needed when the predictive control is implemented.

4.5 Discrete-time MPC with Prescribed Degree of Stability

The closed-loop performance of the predictive control system is specified by the choice of Q and R matrices when a sufficiently large prediction horizon and a large N are used in the design. So often, we select $Q = C^TC$ to minimize the output errors, and R is used to tune the closed-loop response speed. A smaller element in R corresponds to less weight on the corresponding control, hence permitting a larger change in the control increment, and resulting in

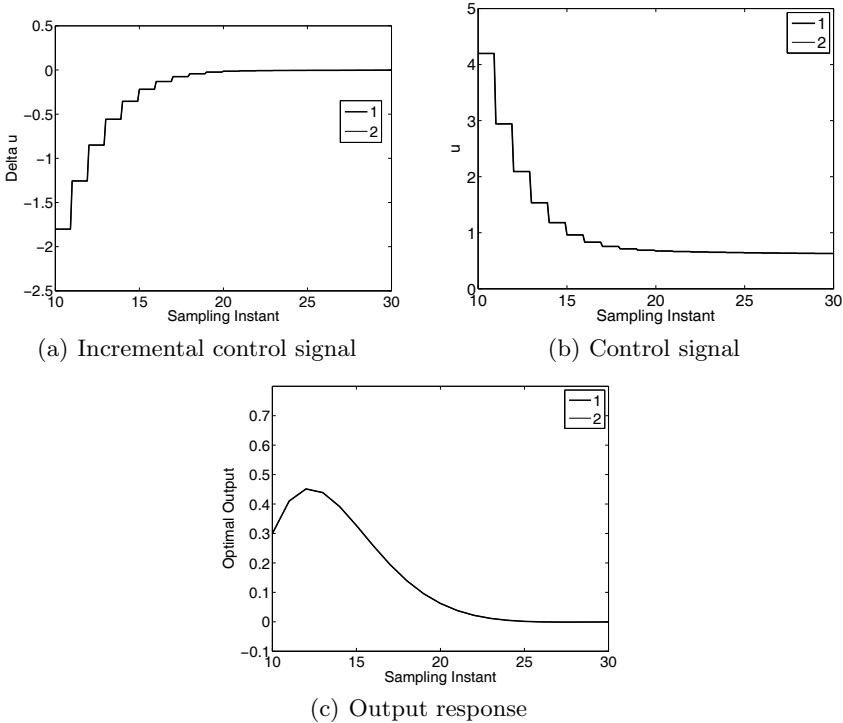


Fig. 4.4. Comparison of responses within one optimization window. Key: line (1) predictive control solution; line (2) DLQR solution with exponential scaling

a faster closed-loop response. For a system with many inputs and outputs, it might be time consuming to tune the closed-loop performance using the Q and R . At other times, it may be desirable to have the closed-loop eigenvalues within a prescribed circle on the complex plane. For instance, by inspection, the three dominant eigenvalues in Example 4.4 are very close to the unit circle (see Figure 4.2). It would be useful if there was a way to move the eigenvalues inside a concentric circle that is well inside the unit circle.

The Design Objective

When constraints are not activated, for a given value of λ , where $0 < \lambda < 1$, the performance specification is to design a predictive control system such that the closed-loop eigenvalues are within the λ circle of the complex plane. In other words, for a given initial condition $x(k_i)$, the predicted future state variable decays at a rate at least equal to λ :

$$\|x(k_i + j | k_i)\| \leq \lambda^j \|x(k_i)\|.$$

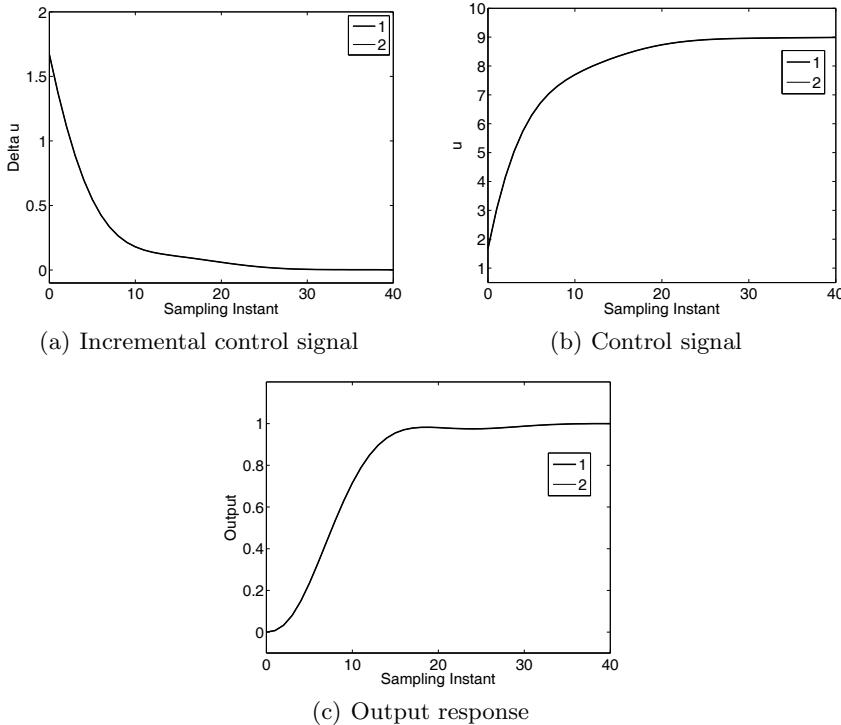


Fig. 4.5. Predictive control without constraints. Key: line (1) Closed-loop control using linear quadratic regulator; line (2) closed-loop predictive control with exponential weight ($\alpha = 1.2$)

The continuous-time counterpart of variable λ is called the degree of stability by Anderson and Moore (1971).

Results for DMPC with Prescribed Degree of Stability

This design objective can be realized using the procedure described in Section 4.4. The results are summarized in the following theorem.

Theorem 4.3. *Subject to the same system state equation*

$$x(k_i + j + 1 \mid k_i) = Ax(k_i + j \mid k_i) + B\Delta u(k_i + j), \quad (4.35)$$

the optimal solution of $\Delta u(k_i + j)$ by minimizing the cost function J_α defined by

$$\begin{aligned} J_\alpha &= \sum_{j=1}^{\infty} \alpha^{-2j} x(k_i + j \mid k_i)^T Q_\alpha x(k_i + j \mid k_i) \\ &\quad + \sum_{j=0}^{\infty} \alpha^{-2j} \Delta u(k_i + j)^T R_\alpha \Delta u(k_i + j) \end{aligned} \quad (4.36)$$

is identical to the solution found by minimizing the original cost:

$$\begin{aligned} J &= \sum_{j=1}^{\infty} \lambda^{-2j} x(k_i + j \mid k_i)^T Q x(k_i + j \mid k_i) \\ &\quad + \sum_{j=0}^{\infty} \lambda^{-2j} \Delta u(k_i + j)^T R \Delta u(k_i + j), \end{aligned} \quad (4.37)$$

where $\alpha > 1$, $0 < \lambda < 1$; Q_α and R_α are defined by

$$\gamma = \frac{\lambda}{\alpha} \quad (4.38)$$

$$Q_\alpha = \gamma^2 Q + (1 - \gamma^2) P_\infty \quad (4.39)$$

$$R_\alpha = \gamma^2 R, \quad (4.40)$$

and P_∞ is the solution of the Riccati equation:

$$\frac{\hat{A}^T}{\gamma} [P_\infty - P_\infty \frac{\hat{B}}{\gamma} (R + \frac{\hat{B}^T}{\gamma} P_\infty \frac{\hat{B}}{\gamma})^{-1} \frac{\hat{B}^T}{\gamma} P_\infty] \frac{\hat{A}}{\gamma} + Q - P_\infty = 0,$$

where the matrices $\hat{A} = \alpha^{-1} A$ and $\hat{B} = \alpha^{-1} B$.

Proof. From the results in Section 4.3 and for $0 < \epsilon < 1$, it is clear that the minimization of the cost function (4.37) will result in,

$$\|x(k_i + j \mid k_i)\| \leq (\lambda \times (1 - \epsilon))^j \|x(k_i)\|, \quad (4.41)$$

based on the same rationale given in (4.14) to (4.17). Because $0 < \lambda < 1$ is specified, (4.41) ensures that the $\|x(k_i + j \mid k_i)\|$ decays, at least at a speed faster than λ^j . In addition, all the closed-loop eigenvalues of the actual system reside inside the circle with a radius of $0 < \lambda < 1$.

Also, from Section 4.3, the optimal control of (4.37) is solvable through the algebraic Riccati equation

$$\frac{A^T}{\lambda} [P_\infty - P_\infty \frac{B}{\lambda} (R + \frac{B^T}{\lambda} P_\infty \frac{B}{\lambda})^{-1} \frac{B^T}{\lambda} P_\infty] \frac{A}{\lambda} + Q - P_\infty = 0. \quad (4.42)$$

We multiply all A and B matrices in (4.42) by the quantity $\frac{\alpha}{\alpha}$ and replacing $\alpha^{-1} A$ by \hat{A} , $\alpha^{-1} B$ by \hat{B} . Denote the quantity $\frac{\lambda}{\alpha} = \gamma$. After that, (4.42) becomes

$$\frac{\hat{A}^T}{\gamma} [P_\infty - P_\infty \frac{\hat{B}}{\gamma} (R + \frac{\hat{B}^T}{\gamma} P_\infty \frac{\hat{B}}{\gamma})^{-1} \frac{\hat{B}^T}{\gamma} P_\infty] \frac{\hat{A}}{\gamma} + Q - P_\infty = 0. \quad (4.43)$$

Multiplying both sides of (4.43) with γ^2 , and letting

$$Q_\alpha = \gamma^2 Q + (1 - \gamma^2) P_\infty \quad (4.44)$$

$$R_\alpha = \gamma^2 R, \quad (4.45)$$

the Riccati equation (4.43) is

$$\hat{A}^T [P_\infty - P_\infty \hat{B} (R_\alpha + \hat{B}^T P_\infty \hat{B})^{-1} \hat{B}^T P_\infty] \hat{A} + Q_\alpha - P_\infty = 0, \quad (4.46)$$

which is the Riccati equation for the cost function (4.36).

Computational Procedure

The design procedure is summarized as below.

1. With a given degree of stability $0 < \lambda < 1$, solve the steady-state Riccati equation for a given $Q \geq 0$, and $R > 0$

$$\frac{A^T}{\lambda} [P_\infty - P_\infty \frac{B}{\lambda} (R + \frac{B^T}{\lambda} P_\infty \frac{B}{\lambda})^{-1} \frac{B^T}{\lambda}] \frac{A}{\lambda} + Q - P_\infty = 0. \quad (4.47)$$

2. Assuming that $\alpha > 1$ is the exponential weighting factor, use the Riccati solution P_∞ from (4.47) in the computation of Q_α and R_α :

$$\gamma = \frac{\lambda}{\alpha} \quad (4.48)$$

$$Q_\alpha = \gamma^2 Q + (1 - \gamma^2) P_\infty \quad (4.49)$$

$$R_\alpha = \gamma^2 R. \quad (4.50)$$

3. Use the Q_α and R_α in the design of model predictive control with the chosen exponential weight α for a sufficiently large N_p

$$\begin{aligned} J = & \sum_{j=1}^{N_p} x(k_i + j | k_i)^T Q_\alpha x(k_i + j | k_i) \\ & + \sum_{j=0}^{N_p} \Delta u(k_i + j)^T R_\alpha \Delta u(k_i + j), \end{aligned} \quad (4.51)$$

based on the pair $(\alpha^{-1} A, \alpha^{-1} B)$.

Example 4.6. Consider the system given in Example 4.4 with identical design specifications except that the degree of stability λ is chosen to be 0.9, namely all closed-loop eigenvalues are specified to be within the circle of radius 0.9.

Solution. With the prescribed degree of stability $\lambda = 0.9$, and the weight matrices Q and R , together with the augmented state model (A, B) (see Example 4.4), we solve the following steady-state algebraic Riccati equation to find P_∞ .

$$\frac{A^T}{\lambda} [P_\infty - P_\infty \frac{B}{\lambda} (R + \frac{B^T}{\lambda} P_\infty \frac{B}{\lambda})^{-1} \frac{B^T}{\lambda}] \frac{A}{\lambda} + Q - P_\infty = 0.$$

The exponential weight factor in Example 4.4 was selected as $\alpha = 1.2$. we calculate γ , Q_α and R_α as

$$\begin{aligned}\gamma &= \frac{\lambda}{\alpha} \\ Q_\alpha &= \gamma^2 Q + (1 - \gamma^2) P_\infty \\ R_\alpha &= \gamma^2 R\end{aligned}$$

With the weight matrices Q_α and R_α and the transformed model (\hat{A}, \hat{B}) , we calculate the predictive control gain matrices. The MATLAB code is given as below.

```
lambda=0.9;
alpha=1.2;
[K,P,E]=dlqr(A/lambda,B/lambda,Q,R);
gamma=lambda/alpha;
Q_alpha=gamma^2*Q+(1-gamma^2)*P;
R_alpha=gamma^2*R;
Ahat=A/alpha;Bhat=B/alpha;
[Omega,Psi]=dmpc(Ahat,Bhat,a,N,Np,Q_alpha,R_alpha);
```

Figure 4.6 shows that all eigenvalues are within the circle of radius 0.9. The condition number of the Hessian matrix is $\kappa(\Omega) = 1080$.

4.6 Tuning Parameters for Closed-loop Performance

There is another option that is available to us for tuning of closed-loop performance. This option relies on the functionality of the Laguerre functions. The basic idea is to use the parameters a and N in the Laguerre functions as part of the closed-loop tuning parameters, where the number of terms N is deliberately limited to a relatively small number.

The question that requires immediate attention is what the control system would be doing if the choice of a and N did not lead to the optimal control specified by the discrete-time LQR system with the pair (Q, R) . The answer is important, particularly if the pair a and N is used as tuning parameters for closed-loop performance. To answer this question, the relationship between the Riccati solution P_∞ and the minimum of the cost function is investigated.

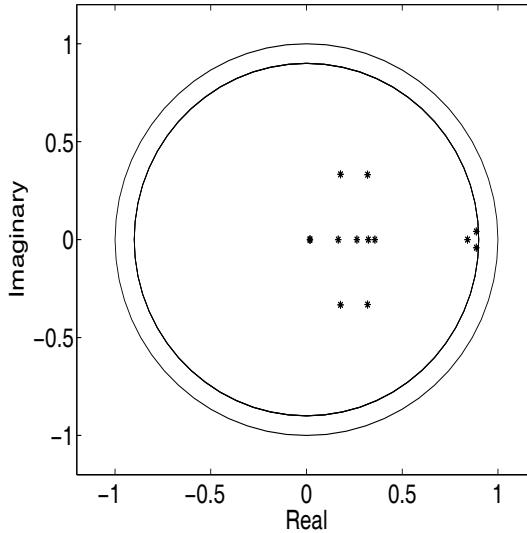


Fig. 4.6. Closed-loop eigenvalues inside λ -circle ($\alpha = 1.2$ and $\lambda = 0.9$)

4.6.1 The Relationship Between P_∞ and J_{min}

Recall from Section 3.3 that the cost function is, after completing the squares

$$\begin{aligned}
 J &= \sum_{j=1}^{N_p} x(k_i + j \mid k_i)^T Q x(k_i + j \mid k_i) \\
 &\quad + \sum_{j=0}^{N_p} \Delta u(k_i + j)^T R \Delta u(k_i + j) \\
 &= (\eta + \Omega^{-1} \Psi x(k_i))^T \Omega (\eta + \Omega^{-1} \Psi x(k_i)) \\
 &\quad - x(k_i)^T \Psi^T \Omega^{-1} \Psi x(k_i) + \sum_{m=1}^{N_p} x(k_i)^T (A^T)^m Q A^m x(k_i), \quad (4.52)
 \end{aligned}$$

where we define

$$\Omega = \left(\sum_{m=1}^{N_p} \phi(m) Q \phi(m)^T + R_L \right) \quad (4.53)$$

$$\Psi = \left(\sum_{m=1}^{N_p} \phi(m) Q A^m \right). \quad (4.54)$$

Therefore, with the optimal solution

$$\eta = -\Omega^{-1} \Psi x(k_i),$$

the minimum of the cost function is

$$J_{min} = x(k_i)^T \left(\sum_{m=1}^{N_p} (A^T)^m Q A^m - \Psi^T \Omega^{-1} \Psi \right) x(k_i). \quad (4.55)$$

Before introducing the exponentially weighted cost function, for a large prediction horizon, it was not appropriate to make a statement about the minimum of the cost function (4.55) because the computation of J_{min} is numerically unstable. This is seen from involvement of the power function of the A matrix, where A has at least an eigenvalue on the unit circle. Thus, in fact, due to the numerical error, J_{min} increases as N_p increases.

With exponential data weighting, J_{min} can be computed with accuracy. Now, it is appropriate to study J_{min} with exponentially decaying weighting in the cost function.

Case A. Sufficiently large N is used

When a sufficiently large N is used in the Laguerre functions, the control trajectory will converge to the underlying optimal control trajectory defined by the discrete-time LQR cost function:

$$\begin{aligned} \tilde{J} &= \sum_{j=0}^{N_p} x(k_i + j | k_i)^T Q x(k_i + j | k_i) \\ &\quad + \sum_{j=0}^{N_p} \Delta u(k_i + j)^T R \Delta u(k_i + j). \end{aligned} \quad (4.56)$$

The minimum of this cost function, with optimal control, is given by

$$\tilde{J}_{min} = x(k_i)^T P_\infty x(k_i). \quad (4.57)$$

With exponential data weighting, in the predictive control, the cost function is

$$\begin{aligned} J &= \sum_{j=1}^{N_p} x(k_i + j | k_i)^T Q_\alpha x(k_i + j | k_i) \\ &\quad + \sum_{j=0}^{N_p} \Delta u(k_i + j)^T R_\alpha \Delta u(k_i + j), \end{aligned} \quad (4.58)$$

subject to the pair $(\alpha^{-1}A, \alpha^{-1}B)$, where $\alpha > 1$ ensures stability of the design model. The weightings Q_α and R_α lead to the equivalence between the weighted and unweighted solutions, as demonstrated before.

Note that when the initial term $x(k_i)^T Q_\alpha x(k_i)$ is added to the cost function for predictive control, $J = \tilde{J}$. Namely, these two cost functions become identical. Although two different approaches, the uniqueness of the minimum of the same quadratic function leads to the relation between the Riccati solution P_∞ and the error matrix of the discrete-time MPC solution as

$$\lim_{N_p \rightarrow \infty} P_{dmpc} = Q_\alpha + \left(\sum_{m=1}^{N_p} (\hat{A}^T)^m Q_\alpha \hat{A}^m - \Psi^T \Omega^{-1} \Psi \right) = P_\infty, \quad (4.59)$$

where $\hat{A} = \alpha^{-1} A$ and $\hat{B} = \alpha^{-1} B$ are used in the computation, and the number of terms N in the Laguerre functions is assumed to be sufficiently large. We emphasize that this relationship is established based on the choice of a sufficiently large N so that the discrete-time MPC control trajectory converges to the underlying control trajectory of the discrete-time LQR system. The relation is illustrated by the following example.

Example 4.7. Consider the continuous-time plant model with transfer function

$$G(s) = \begin{bmatrix} \frac{12.8(-s+4)^2}{(16.7s+1)(s+4)^2} & \frac{-1.89}{(21.0s+1)} \\ 0 & \frac{-19.4(-3s+4)^2}{(14.4s+1)(3s+4)^2} \end{bmatrix}. \quad (4.60)$$

Choosing sampling interval $\Delta t = 1$, design a predictive control system with integral action. Assume that $Q = C^T C$, and $R = I$; $a = 0$, $N_p = 140$, and compute the solution using the long prediction horizon with exponential data weight $\alpha = 1.2$, and modified Q_α and R_α . Compare the relative errors of the diagonal elements between Riccati solution P_∞ and the matrix P_{dmpc} when $N_1 = N_2 = 6$ and $N_1 = N_2 = 8$.

Solution. With the choice of weight matrices Q and R , we compute the Riccati solution P using the MATLAB `dlqr` function. Then, we obtain the weight matrices in the cost function of the predictive control as Q_α and R_α . The matrix P_{dmpc} is computed as

$$P_{dmpc} = Q_\alpha + \left(\sum_{m=1}^{N_p} (\hat{A}^T)^m Q_\alpha \hat{A}^m - \Psi^T \Omega^{-1} \Psi \right).$$

Define the j th element of the relative error:

$$e(j) = |P_\infty(j, j) - P_{dmpc}(j, j)| / P_\infty(j, j),$$

where the elements $P_\infty(j, j)$ and $P_{dmpc}(j, j)$ are the diagonal elements of P_∞ and P_{dmpc} , respectively. The error vector $e(.)$ is for $N_1 = N_2 = 6$,

$$e = 10^{-3} \times [0.0929 \ 0.2126 \ 0.2289 \ 0.5548 \ 0.1047 \ 0.0546 \ 0.3086 \ 0.0451 \ 0.0048].$$

For comparison, with $N_1 = N_2 = 8$, the error vector $e(.)$ is

$$e = 10^{-3} \times [0.0100 \ 0.0140 \ 0.0038 \ 0.2477 \ 0.0441 \ 0.0307 \ 0.1533 \ 0.0090 \ 0.0008].$$

Therefore, with an increasing N , the relative errors between the diagonal elements of these two matrices are reduced. Furthermore, by choosing $x(k_i)$ being a vector containing unity elements, the minimum of the cost function is evaluated. For discrete-time LQR, $J_{min} = 11.0811$, and for the discrete-time MPC when $a_1 = a_2 = 0.0$, $N_1 = N_2 = 8$, $J_{min} = 11.0812$.

Case B. Relatively small N is used

This is to investigate what occurs when a relatively smaller N is used for a given a . When this happens, the predictive control trajectory will not converge to the underlying optimal control trajectory defined by the pair of weight matrices (Q, R) . If we term the case when N is sufficiently large as the global optimum for the given cost function, then the case corresponding to a smaller N could be termed a truncated approximation to the global optimum. There is only one global optimal solution once Q and R are selected. However, there are many approximations to the optimal solutions depending on the selection of the parameters a and N in the Laguerre functions. They provide the user with the means to select the closed-loop performance that might be desirable in a specific application. More explicitly, once Q and R are selected, the parameters a and N are used as fine-tuning parameters for the closed-loop performance. This is particularly useful when dealing with a complex system, where the variations of a and N are selected for each input independently to find the desired closed-loop performance.

An approximation to the global optimal solution could also be interpreted as a global optimal solution on its own for a pair of weight matrices \tilde{Q} and \tilde{R} which are unknown, also different from the original Q and R . This interpretation also sets the conditions such that the approximate optimal solution will stabilize the closed-loop system. This is obtained through reverse engineering.

Suppose that the pair (or pairs) of a and N parameters are chosen to form the Laguerre functions, the cost function J is minimized with respect to optimal control. Then the minimum of the cost function J is

$$J_{min} = x(k_i)^T \left(\sum_{m=1}^{N_p} (\hat{A}^T)^m Q_\alpha \hat{A}^m - \Psi^T \Omega^{-1} \Psi \right) x(k_i), \quad (4.61)$$

where $\hat{A} = \alpha^{-1} A$ and $\hat{B} = \alpha^{-1} B$ are used in the computation. This J_{min} is unique with respect to the choice of a and N parameters, when Ω^{-1} exists. The second term $x(k_i)^T \Psi^T \Omega^{-1} \Psi x(k_i)$ is a function of the a and N parameters. With restricted a and N , this J_{min} is different from the global minimum, and the optimal control is different from the discrete-time LQR optimal control defined by the pair (Q, R) . Therefore, for a restricted pair of a and N parameters, there is a pair of unknown weight matrices Q and R defining a different cost function. We call the unknown weight matrices \tilde{Q} and \tilde{R} . Through reverse engineering, we find out what they are.

With the original choice of Q and R , and a , N parameters, the Riccati solution P_{dmpc} is calculated as

$$P_{dmpc} = Q_\alpha + \left(\sum_{m=1}^{N_p} (\hat{A}^T)^m Q_\alpha \hat{A}^m - \Psi^T \Omega^{-1} \Psi \right), \quad (4.62)$$

where $\alpha > 1$ is used to ensure that \hat{A} is stable. Assuming that the choice of a and N leads to a stable closed-loop system, then following the derivation of linear quadratic optimal control, the Riccati equation holds

$$\hat{A}^T [P_{dmpc} - P_{dmpc} \hat{B} (\tilde{R}_\alpha + \hat{B}^T P_{dmpc} \hat{B})^{-1} \hat{B}^T P_{dmpc}] \hat{A} + \tilde{Q}_\alpha - P_{dmpc} = 0. \quad (4.63)$$

Choosing $\tilde{R}_\alpha = R_\alpha$, then the free parameter \tilde{Q}_α can be used to balance the Riccati equation, leading to

$$\tilde{Q}_\alpha = P_{dmpc} - \hat{A}^T [P_{dmpc} - P_{dmpc} \hat{B} (R_\alpha + \hat{B}^T P_{dmpc} \hat{B})^{-1} \hat{B}^T P_{dmpc}] \hat{A}. \quad (4.64)$$

This means that by choosing a restricted pair of a and N parameters, the predictive control system is equivalent to a discrete-time LQR system with a pair of weight matrices \tilde{Q} and \tilde{R} . By substituting the value of \tilde{Q}_α , the value of \tilde{Q} is

$$\begin{aligned} \tilde{Q} &= \alpha^2 \tilde{Q}_\alpha - (\alpha^2 - 1) P_{dmpc} \\ &= P_{dmpc} - A^T [P_{dmpc} - P_{dmpc} B (R + B^T P_{dmpc} B)^{-1} B^T P_{dmpc}] A. \end{aligned} \quad (4.65)$$

Therefore, for a small N , the closed-loop predictive control system is stable if P_{dmpc} is positive definite and \tilde{Q} is non-negative definite, where \tilde{Q} is calculated using (4.65). There are a few comments to make. Firstly, (4.65) says that if P_{dmpc} is equal to P_∞ from the original cost function using discrete-time LQR design, then $\tilde{Q} = Q$, thus there is no change in the cost function. However, if P_{dmpc} differs from P_∞ , then equivalently a different discrete-time LQR problem is solved using the predictive control framework, with the pair \tilde{Q} and \tilde{R} . Additionally, the prescribed degree of stability λ can be effectively enforced with an arbitrary pair of a and N , without \tilde{Q} entering the computation.

It is worthwhile to emphasize that \tilde{Q} will not enter the computation, and its existence is for theoretical justification and for understanding the essence of the problem in relation to the existing discrete-time LQR design.

Example 4.8. Suppose that a first-order plus time-delay system is described by the transfer function

$$G(z) = \frac{0.1z^{-6}}{z - 0.8}.$$

Choosing $Q = C^T C$, and $R = 0.1$, $\alpha = 1.2$ as the design parameters, show the variation of closed-loop performance by varying the Laguerre pole a for $0 \leq a \leq 0.9$ where the parameter $N = 1$ is fixed. The prediction horizon $N_p = 46$ is selected for the computation.

Solution. To illustrate the tuning procedure, a unit step response test is used with zero initial condition of the state variables. We select the value of a equal to 0, 0.3, 0.6, 0.9, respectively, in the predictive control system and compare the closed-loop control results with the results obtained using DLQR design. Figure 4.7 shows the comparative results. It is seen that for $N = 1$, the closed-loop predictive control system is stable for the range of a used in the design. Furthermore, the optimal DLQR system offers the fastest rise time and slight over-shoot. For this particular system, as a increases, the closed-loop response speed of the predictive control system reduces. There is a performance trend dependent on the variation of a .

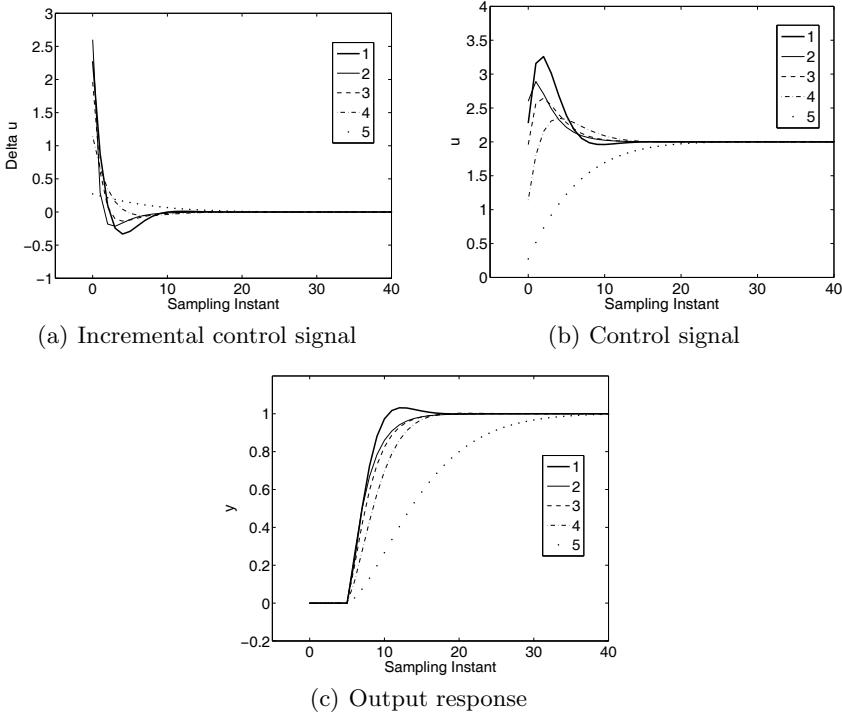


Fig. 4.7. Tuning of predictive control system ($N = 1$, varying a). Key: line (1) DLQR control; line (2) $a = 0$; line (3) $a = 0.3$; line (4) $a = 0.6$; line (5) $a = 0.9$

4.6.2 Tuning Procedure Once More

With exponential data weighting, the closed-loop performance parameters are very similar to the discrete-time LQR performance parameters. Basically, the choice of weight matrices Q and R determines the closed-loop performance. The prediction horizon N_p no longer plays a role in the design, because a large N_p is used to approximate an infinite prediction horizon. For a choice of large N , with any $0 \leq a < 1$, the trajectory of the future control trajectory converges to the underlying optimal control trajectory defined by the corresponding LQR control law. Therefore, if we wish, a large N is selected for each input variable, so that the closed-loop performance is solely dependent on the Q and R matrices. Of course, the prescribed degree of stability, λ , is a very important parameter in the specification of closed-loop performance. This is because, with this parameter, the closed-loop poles of the predictive control system are constrained to be within a λ -circle of the complex plane, where $0 < \lambda < 1$.

It is understandable that tuning closed-loop performance through the choice of Q and R matrices can be quite time consuming. There are two

reasons for this. One is that not only the diagonal elements, but also the off-diagonal elements in these matrices influence the closed-loop performance. The other reason is that the diagonal and off-diagonal elements lack a direct link to the closed-loop performance, hence offering little by way of intuitive guidelines to the designer. It goes without saying that tuning a predictive control system with many inputs, many outputs and many state variables is indeed a complex task in its own right.

What we propose here is to make the tuning procedure as simple as possible. First, the weight matrix Q is always important for the closed-loop performance. However, the optimal Q is often selected as

$$Q = C^T C.$$

With integrators embedded in the MPC design, C is the output matrix of the augmented model so it contains all zeros apart from the unit diagonal elements corresponding to the outputs. With this choice, the closed-loop eigenvalues are determined by the weight matrix R . More specifically, if it is a single-input and single-output system where $R = r_w$ is a scalar, the closed-loop eigenvalues are the inside-the-unit-circle zeros of the equation,

$$1 + \frac{1}{r_w} \frac{G_m(z)G_m(z^{-1})}{(z-1)(z^{-1}-1)} = 0. \quad (4.66)$$

In the multi-input and multi-output case, when the weighting matrices $Q = C^T C$, and $R = r_w I$, the closed-loop poles are the inside-the-unit-circle zeros of the equation,

$$\det(I + \frac{1}{r_w} \frac{G_m(z)G_m(z^{-1})^T}{(z-1)(z^{-1}-1)}) = 0,$$

where $G_m(z) = C_m(zI - A_m)^{-1}B_m$ is the z-transfer function for the plant. By varying the scalar r_w , we will obtain the set of closed-loop eigenvalues as a function of the weight r_w , and choose the weight r_w that correspond to the desired closed-loop eigenvalues.

A more general case, but still maintaining simplicity, will include the choice of Q being $Q = C^T Q_y C$, where $Q_y > 0$ is the diagonal weight matrix that the designer wishes to use for distributing weights on individual output variables, as well as the choice of $R > 0$ being a diagonal matrix for distributing weights on individual input variables. Note that weighting R is assumed to be a diagonal matrix to simplify the computation of MPC using the Laguerre basis functions. The smaller elements in R corresponds to faster closed-loop response speed.

Next, the exponential weight factor α needs to be specified. Use of the exponential weight will avoid the numerical ill-conditioning problem for the class of MPC systems that have embedded integrators. If the plant is stable, any $\alpha > 1$ will serve this purpose. A modest α is recommended when dealing

with constraints. For instance, $\alpha = 1.1$ is sufficient for the class of stable plants or plants with integrators. When using exponential data weighting the closed-loop performance requires to be compensated so that the original targeted performance by Q and R remains unchanged. This compensation is simply achieved by using

$$\gamma = \frac{1}{\alpha} \quad (4.67)$$

$$Q_\alpha = \gamma^2 Q + (1 - \gamma^2) P_\infty \quad (4.68)$$

$$R_\alpha = \gamma^2 R, \quad (4.69)$$

where P_∞ is the Riccati solution of the algebraic Riccati equation with the original augmented model (A, B) , and the performance matrices Q and R . If desired, a prescribed degree of stability λ can also be embedded at this stage, where the value of γ is selected to be $\gamma = \frac{\lambda}{\alpha}$.

With Q , R and λ chosen, the closed-loop performance of the predictive control system is determined. This closed-loop performance will be achieved when the number of terms in the Laguerre functions is selected to be large. How large is dependent on the selection of the scaling factor a . Bearing this in mind, the pair of a and N is used as fine-tuning parameters for the closed-loop performance.

The tuning procedure is summarized as follows.

1. Select the weight matrices $Q = C^T Q_y C$ and R , where $Q_y \geq 0$, $R > 0$ are diagonal matrices with elements corresponding to the weights on the individual output and control signals, respectively. A larger element in Q_y means the demanding of a faster response from that particular output, while a larger element in R means less control action required from that particular control signal.
2. Specify an $\alpha > 1$ to ensure numerical stability and a λ -circle in which all the closed-loop poles of the predictive control system are to reside.
3. Use a large prediction horizon N_p to approximate the infinite horizon control case, and set the Laguerre function order N to be a large value, and the Laguerre pole a to be close to the dominant pole of the closed-loop LQR system.
4. Calculate the Ω and Ψ in the cost function of the predictive control system:

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(k_i). \quad (4.70)$$

5. Increase the Laguerre function order N until the closed-loop performance has no further change. This is the predictive control system that is identical to the DLQR system with a prescribed degree of stability.
6. If we wish to fine-tune the closed-loop performance, we could reduce the Laguerre function order N to find the approximations to the optimum. We could also perturb the Laguerre pole a to vary the closed-loop performance for some small N . These variations are equivalent to different choices of Q matrix.

4.7 Exponentially Weighted Constrained Control

This section examines constrained control with exponential weighting in the cost function. Recall in the solution of exponentially weighted predictive control problem, essentially, the cost function

$$\begin{aligned} J = & \sum_{j=1}^{N_p} \hat{x}(k_i + j | k_i)^T Q_\alpha \hat{x}(k_i + j | k_i) \\ & + \sum_{j=0}^{N_p} \Delta \hat{u}(k_i + j)^T R_\alpha \Delta \hat{u}(k_i + j) \end{aligned} \quad (4.71)$$

is minimized using the pair $(\alpha^{-1}A, \alpha^{-1}B)$, where $\alpha > 1$ is used to scale the eigenvalues of matrix A . The equivalence to the optimal DLQR solution was also established. Suppose that with Q and R weight matrices, $\Delta u(k_i + j)$ and $x(k_i + j | k_i)$ are the LQR solution, then, the transformed variables are

$$\Delta \hat{u}(k_i + j) = \alpha^{-j} \Delta u(k_i + j); \quad \hat{x}(k_i + j | k_i) = \alpha^{-j} x(k_i + j | k_i).$$

As the constraints are specified in terms of the original physical variables, the constraints need to be transformed to correspond with the exponentially weighted variables in the solution of the optimization problem. Since the transformed variables at the beginning of the optimization window are $\Delta \hat{u}(k_i) = \Delta u(k_i)$ and $\hat{u}(k_i) = u(k_i)$, and if the constraints are only imposed at the initial sample of the variables within the window, then the constraints will be used unchanged, namely, the constraints are formulated as

$$\Delta u^{\min} \leq L(0)^T \eta \leq \Delta u^{\max}; \quad u^{\min} \leq u(k_i - 1) + L(0)^T \eta \leq u^{\max}.$$

However, because the constraints on the state variables are imposed one step ahead, the exponential factor needs to be included, leading to

$$\alpha^{-1} X^{\min} \leq \alpha^{-1} A x(k_i) + \phi(0)^T \eta \leq \alpha^{-1} X^{\max}.$$

Suppose that all the original constraints are uniformly imposed on the future samples. Then the transformed constraints are changing with respect to the future sample instant. For example, the future constraints for Δu at $j = 1, 2, 3, \dots, m$ become

$$\begin{aligned} \alpha^{-1} \Delta u^{\min} & \leq L(1)^T \eta \leq \alpha^{-1} \Delta u^{\max} \\ \alpha^{-2} \Delta u^{\min} & \leq L(2)^T \eta \leq \alpha^{-2} \Delta u^{\max} \\ \alpha^{-3} \Delta u^{\min} & \leq L(3)^T \eta \leq \alpha^{-3} \Delta u^{\max} \\ & \vdots \\ \alpha^{-m} \Delta u^{\min} & \leq L(m)^T \eta \leq \alpha^{-m} \Delta u^{\max}. \end{aligned} \quad (4.72)$$

The constraints on the state variable are transformed to

$$\begin{aligned} \alpha^{-2} X^{min} &\leq \alpha^{-2} A^2 x(k_i) + \phi(1)^T \eta \leq \alpha^{-2} X^{max} \\ \alpha^{-3} X^{min} &\leq \alpha^{-3} A^3 x(k_i) + \phi(2)^T \eta \leq \alpha^{-3} X^{max} \\ &\vdots \\ \alpha^{-m} X^{min} &\leq \alpha^{-m} A^m x(k_i) + \phi(m-1)^T \eta \leq \alpha^{-m} X^{max}, \end{aligned} \quad (4.73)$$

where $\phi(j)$ is the data vector used in the design of predictive control. As for the constraints on the control variable, they are transformed to:

$$\begin{aligned} u^{min} &\leq u(k_i - 1) + \alpha L(1)^T \eta \leq u^{max} \\ u^{min} &\leq u(k_i - 1) + (\alpha L(1)^T + \alpha^2 L(2)^T) \eta \leq u^{max} \\ u^{min} &\leq u(k_i - 1) + (\alpha L(1)^T + \alpha^2 L(2)^T + \alpha^3 L(3)^T) \eta \leq u^{max} \\ &\vdots \\ u^{min} &\leq u(k_i - 1) + \sum_{j=1}^m \alpha^j L(j)^T \eta \leq u^{max}. \end{aligned} \quad (4.74)$$

There are a few comments to be made here. It is seen here that the originally uniformly imposed constant constraints are all transformed into constraints that are functions of the exponential weight factor α^{-j} . Because α is greater than unity, the bounds converge to some constants as the future sample index increases. On the other hand, if the original constraints are not uniformly imposed, and if they are imposed in an exponentially increasing manner with the factor α , then the constraints become constants for the transformed variables, at least for Δu and state variable x . In addition, because the control trajectory $\hat{u}(k_i + j)$ is captured by a set of Laguerre functions, which are exponentially decaying functions, it decays as future sample index j increases.

Example 4.9. Suppose that a second-order discrete-time system with time delay is described by

$$G(z) = \frac{z^{-16}}{(z - 0.8)(z - 0.6)}.$$

Design and simulate a predictive control system with unit step input and zero initial conditions. The weight matrices are $Q = C^T C$, $R = 0.1$. With $N = 8$, $a = 0.5$, $N_p = 46$ the predictive control system is almost identical to the DLQR control system. The constraints are specified as

$$-0.3 \leq \Delta u(k) \leq 1; \quad 0.7 \leq u(k) \leq 1.$$

For the choice of $\alpha = 1$, $\alpha = 1.1$ and $\alpha = 1.2$, illustrate the closed-loop performance and numerical condition of the Hessian matrix with respect to the exponential weight factor α .

Table 4.3. Condition number of Hessian matrix Ω

α	1	1.1	1.2
$\kappa(\Omega)$	4005	538	141

Solution. The condition numbers for the Hessian matrix are listed in Table 4.3. For this simple system, without exponential data weighting ($\alpha = 1$), the condition number of the Hessian is very large (4005). With the choice of $\alpha = 1.1$, the condition number is reduced to 13% of the previous case. When α further increases to 1.2, the condition number reduces to 141. In addition, with $\alpha = 1.1$, the Ω and Ψ matrices become invariant with the large prediction horizon N_p . For comparison purpose, the unconstrained case is shown in Figure 4.8.

To impose constraints, we transform the constraints to the forms that embed the exponential factor α using (4.72) and (4.74). We impose the con-

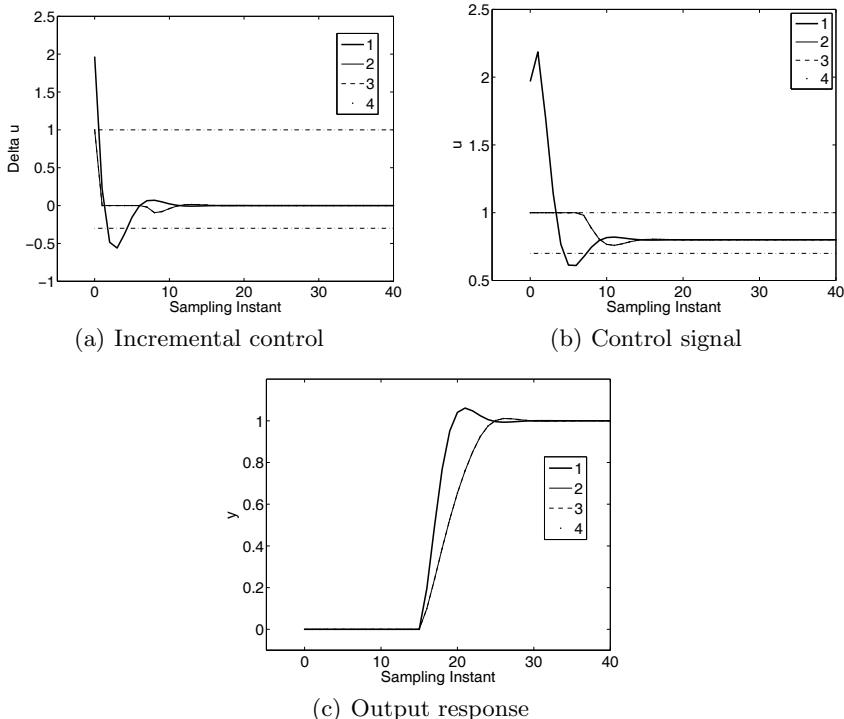


Fig. 4.8. Predictive control with constraints. Key: line (1) without constraints; line (2) constraints on the first sample ($\alpha = 1.2$); line (3) constraints on the first 10 samples ($\alpha = 1.2$); line (4) constraints on the first 10 samples ($\alpha = 1.1$). The plots for case 2, 3, 4 are identical

straints for the first 10 samples. All together, there are three cases being investigated, where $\alpha = 1.2$ with constraints on the first sample and the first 10 samples, and $\alpha = 1.1$ with the first 10 samples. Although different α values are used in the constrained control simulation, the numerical results show that the constrained control results are identical, as seen in Figure 4.8. These results mean that when transforming the constraints to the forms that embed the exponential weight α , the constrained control system is consistent with respect to the weight factor α .

To emphasize this, use of exponential data weighting significantly improves the numerical condition of the predictive control system, and removes the effect of prediction horizon in the design. Also, a prescribed degree of stability can be added in the design.

4.8 Additional Benefit

The following example will illustrate the additional benefit when using exponential data weighting in the predictive control design.

Example 4.10. A mechanical system (van Donkelaar *et al.*, 1999) that is highly oscillatory and non-minimum-phase, is described by

$$G(z) = \frac{-5.7980z^3 + 19.5128z^2 - 21.6452z + 7.9547}{z^4 - 3.0228z^3 + 3.8630z^2 - 2.6426z + 0.8084}. \quad (4.75)$$

This system has four poles located at

$$0.5162 \pm j0.7372 \quad 0.9952 \pm j0.0877$$

and the three zeros at

$$1.3873 \quad 0.9891 \pm j0.1034.$$

The unit step response and the frequency response of this system are shown in Figure 4.9. Design and simulate predictive control systems with and without exponential data weighting ($\alpha = 1.2$ and $\alpha = 1$); and compare the results. The design specification for the weight matrices are $Q = C^T C$; $R = 0.3$. The prediction horizon, and the parameters in the Laguerre functions will be selected depending on the choice of α . Observe the large differences between N and N_p when $\alpha = 1$ and $\alpha = 1.2$.

Solution.

The LQR control system

The discrete-time LQR solution leads to the gain vector

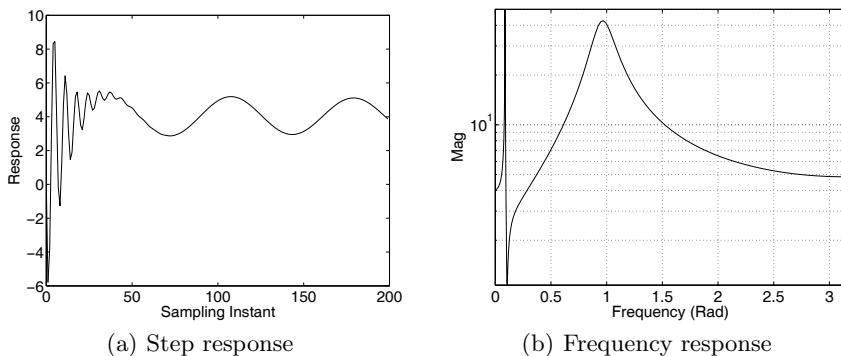


Fig. 4.9. Open-loop response of mechanical system

$$K = [2.0269 \quad -4.8132 \quad 3.6065 \quad -0.8047 \quad 0.1233].$$

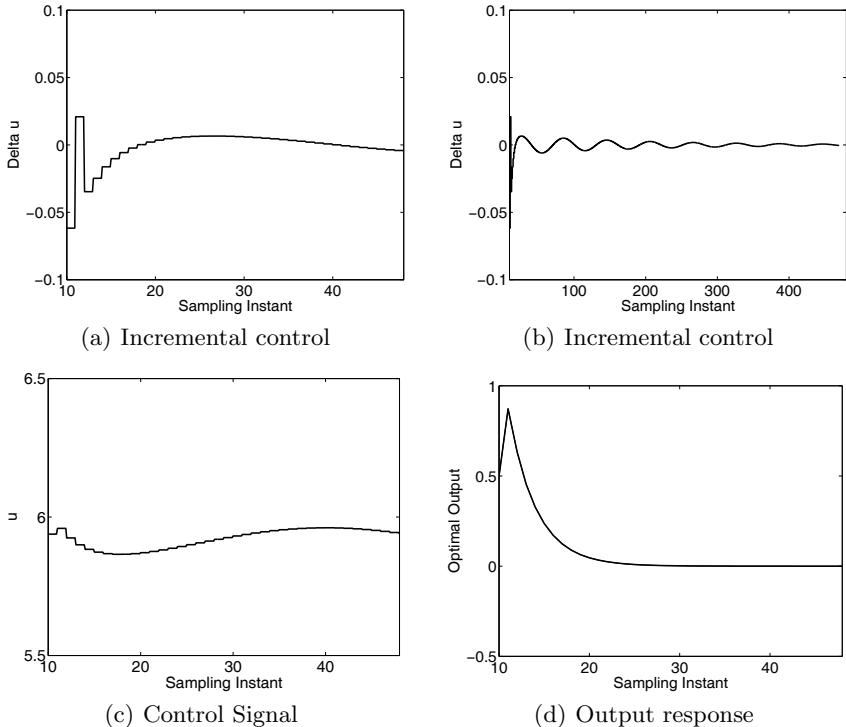
Assuming the initial state variable condition at sampling time $k_i = 10$ as

$$x(k_i) = [0.1 \quad 0.2 \quad 0.3 \quad 0.4 \quad 0.5],$$

the LQR optimal trajectory $\Delta u(k_i + j)$ is constructed using the $x(k_i)$, A , B and K , and is shown in Figure 4.10a for the initial response and Figure 4.10b for the entire response. It is seen that because the system is highly oscillatory, it takes about 460 samples for the LQR control trajectory to decay to zero. In addition, the initial part of the control trajectory is quite complicated (see Figure 4.10a). However, it takes about 20 samples for the system output to decay to zero (see Figure 4.10d), which illustrates that the closed-loop system has a very fast response speed.

Predictive control without exponential weighting ($\alpha = 1$)

From Figure 4.10, we see that in order for the predictive control system to match the discrete-time LQR system, the prediction horizon needs to be selected as 460 or above. As for the parameters in the Laguerre function, if $a = 0$, then N needs to be 400 or above to achieve the LQR desired results. If a larger a is used, then the parameter N is smaller. Indeed, with the selection of $a = 0.9$ and $N = 60$, and predictive control system matches the results obtained from discrete-time LQR within one optimization window. In contrast, if $a = 0$ and $N = 199$ are selected, then the predictive control signals will match the LQR control up to the sampling instant of 199. Figure 4.11 shows the comparison results with one optimization with $k_i = 10$ and the control signal before the optimization window is $u(9) = 6$. Control increment Δu continues to oscillate about 500 samples, which is not shown in Figure 4.11a. It is worthwhile to mention that the Hessian matrix is ill-conditioned with either $a = 0.9$ or $a = 0$. When $a = 0.9$, the condition number is 4.6595×10^5 , while

**Fig. 4.10.** LQR control of mechanical system

the condition number is 1.7418×10^4 for the case $a = 0$. The question arises as why the MPC with Laguerre functions ($a = 0.9$) has the larger condition number. This is because the Laguerre functions with $a = 0.9$ did not decay to zero after 199 samples and the convolution sum $\phi(m)$ continued until m reached $N_p = 480$. In comparison, when $a = 0$, the incremental control is zero after 199 samples.

Predictive control with exponential weighting ($\alpha = 1.2$)

With $\alpha = 1.2$, the original variables $\Delta u(k_i + j)$ and $x(k_i + j | k_i)$ are transformed into $\hat{\Delta}u(k_i + j) = \alpha^{-j}\Delta u(k_i + j)$ and $\hat{x}(k_i + j | k_i) = \alpha^{-j}x(k_i + j | k_i)$. The transformed optimal control trajectory is shown in Figure 4.12. It is seen (Figure 4.12a) that the slow oscillation from the original Δu trajectory has disappeared. It is a relatively easy task to model the transformed optimal control trajectory using the Laguerre functions. In addition, the prediction horizon is much less than 480, instead, 46 is selected. Thus, with $N_p = 46$, $N = 10$ and $a = 0.4$, the predictive control system matches the discrete-time LQR system as shown in Figure 4.12. The condition number of the Hessian

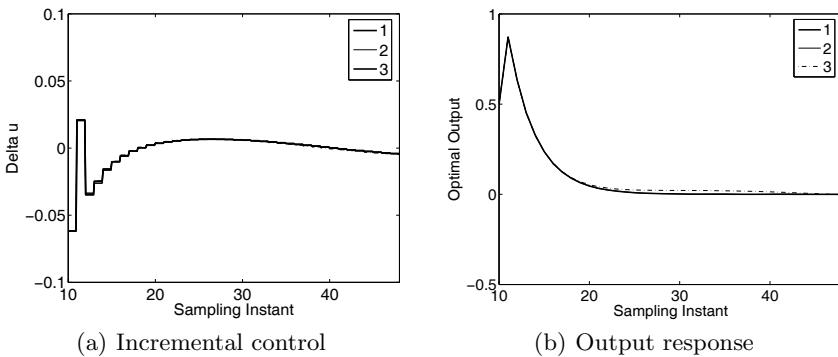


Fig. 4.11. Comparison study: LQR vs predictive control. Key: line (1) LQR; line (2) predictive control $a = 0.9$ and $N = 60$; line (3) predictive control $a = 0$ and $N = 199$.

is 59.2435, which is contrasted with the condition number 4.6595×10^5 when $\alpha = 1$ (no exponential weighting) and $a = 0.9$. Also, the feedback gain matrix is

$$K_{mpc} = [2.0269 \quad -4.8131 \quad 3.6065 \quad -0.8047 \quad 0.1233],$$

which is identical to K with accuracy up to 4 digits.

Constrained control ($\alpha = 1.2$)

Assume that the constraints are specified as

$$-0.01 \leq \Delta u(k) \leq 0.05; \quad 0.1 \leq u(k) \leq 0.4.$$

A unit step signal is used as the reference signal at $k = 0$. Figure 4.13 shows the closed-loop control system responses. It is seen that the constraint on

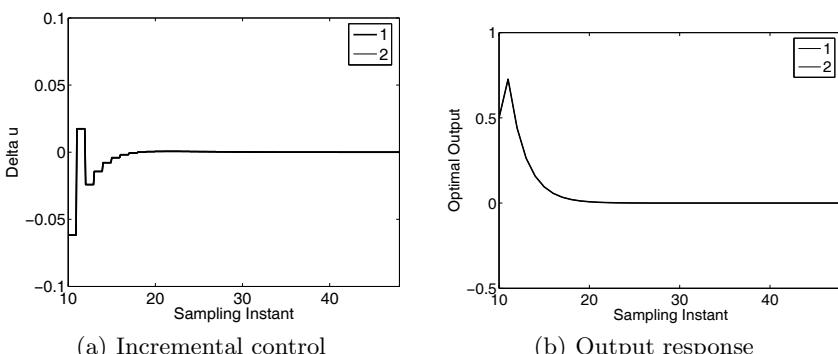


Fig. 4.12. Comparison results: LQR vs predictive control. Key: line (1) LQR; line (2) predictive control $a = 0.4$ and $N = 10$ ($\alpha = 1.2$).

$0.1 \leq u$ was not met at the same time when the constraint on $\Delta u(k)$ became activated. When both of them are activated, they become conflict constraints. Nevertheless, Hildreth's quadratic programming method found a compromise solution where the constraint on u was relaxed.

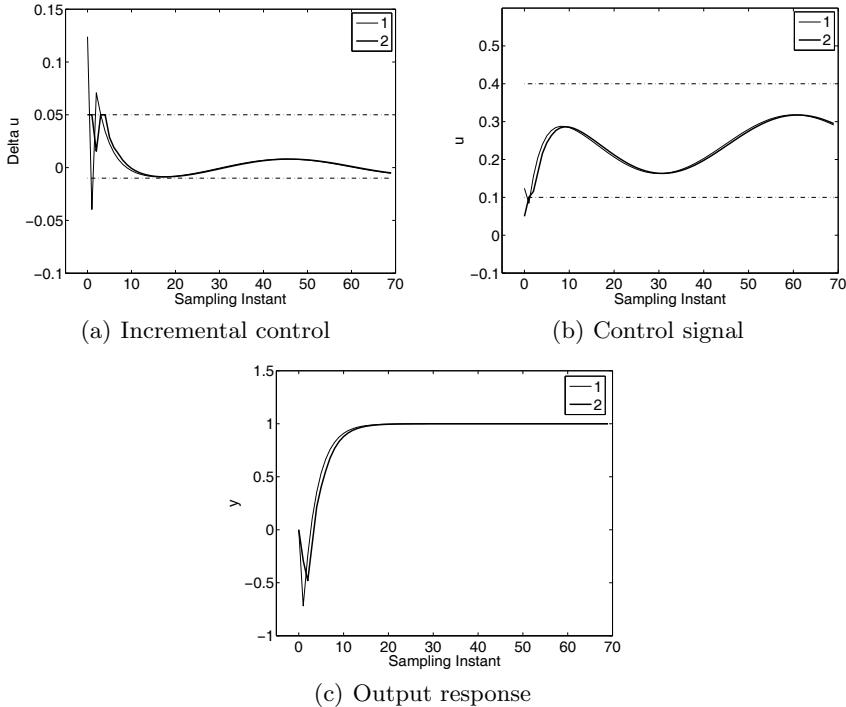


Fig. 4.13. Predictive control with constraints. Key: line (1) without constraints; line (2) constraints on the first sample ($\alpha = 1.2$)

4.9 Summary

This chapter has proposed use of exponential data weighting in the design of predictive control. There are two main purposes for using exponential data weighting in the design. For the class of model predictive control systems that have embedded integrator(s) in the design model, the prediction horizon N_p is limited to a finite value because the design model has at least one eigenvalue on the unit circle. With the exponential data weighting, the design model is modified so that it has all eigenvalues within the unit circle of the complex plane, thus prediction is based on a stable model. As a result, the prediction horizon N_p can be selected to be sufficiently large so as to approximate the

infinite horizon case. Furthermore, the numerical ill-conditioning issue with regard to a large prediction horizon without exponential weighting is removed. This is practically very important.

The results of this chapter followed three stages.

1. In stage one, by choosing a cost function with exponentially decreasing weight α^{-j} , $j = 0, 1, 2, \dots, N_p$, where $\alpha > 1$, the original predictive control problem is converted into one with transformed variables \hat{x} and $\Delta\hat{u}$. That is, to minimize

$$J = \sum_{j=1}^{N_p} \hat{x}(k_i + j | k_i)^T Q \hat{x}(k_i + j | k_i) + \sum_{j=0}^{N_p} \Delta\hat{u}(k_i + j)^T R \Delta\hat{u}(k_i + j),$$

$$\hat{x}(k+1) = \frac{A}{\alpha} \hat{x}(k) + \frac{B}{\alpha} \Delta\hat{u}(k),$$

subject to transformed constraints. This very simple modification of the original model leads to a stable design model and allows a large prediction horizon to be used in the design. Hence, the computation of the predictive control gain matrices is no longer sensitive to the choice of prediction horizon. One uncertain aspect of this design is that depending on the value of α , the closed-loop predictive control system may not be stable, particularly in the situation where α is large. This drawback is overcome in the stage-two development.

2. To guarantee that the predictive control system is stable when using exponentially decreasing weight, we need to introduce a different pair of weight matrices Q_α and R_α , where P_∞ is the steady-state Riccati solution of the original cost function, and

$$\gamma = \frac{1}{\alpha} \quad (4.76)$$

$$Q_\alpha = \gamma^2 Q + (1 - \gamma^2) P_\infty \quad (4.77)$$

$$R_\alpha = \gamma^2 R, \quad (4.78)$$

then we use the Q_α and R_α in the design of model predictive control with the chosen exponential weight α for a sufficiently large N_p by minimizing

$$J = \sum_{j=1}^{N_p} \hat{x}(k_i + j | k_i)^T Q_\alpha \hat{x}(k_i + j | k_i) + \sum_{j=0}^{N_p} \Delta\hat{u}(k_i + j)^T R_\alpha \Delta\hat{u}(k_i + j), \quad (4.79)$$

based on the pair $(\alpha^{-1}A, \alpha^{-1}B)$. This new pair of Q_α and R_α will guarantee that the exponentially weighted solution will lead to the identical solutions of the actual $x(k_i + j | k_i)$ and $\Delta u(k_i + j)$.

3. In stage-three development, a prescribed degree of stability is embedded into the design where the actual $x(k_i + j | k_i)$ and $\Delta u(k_i + j)$ are dictated to decay at a rate not less than $\lambda^j ||x(k_i)||$, $\lambda < 1$. Alternatively, all the eigenvalues of the predictive control system are strictly within the λ circle. This is achieved by solving the Riccati equation to find P_∞ :

$$\frac{A^T}{\lambda} [P_\infty - P_\infty \frac{B}{\lambda} (R + \frac{B^T}{\lambda} P_\infty \frac{B}{\lambda})^{-1} \frac{B^T}{\lambda} P_\infty] \frac{A}{\lambda} + Q - P_\infty = 0, \quad (4.80)$$

and choosing the pair of weight matrices Q_α and R_α according to

$$\gamma = \frac{\lambda}{\alpha} \quad (4.81)$$

$$Q_\alpha = \gamma^2 Q + (1 - \gamma^2) P_\infty \quad (4.82)$$

$$R_\alpha = \gamma^2 R. \quad (4.83)$$

When using exponential data weighting, the optimization is performed using the transformed variables and the transformed variables have a faster decay rate than the original variables. As a result, the prediction horizon is shorter and the number of terms in the Laguerre functions is smaller. These will result in a less number of constraints to be imposed on the future samples in a constrained control environment. Additional benefit to predictive control of complex system is illustrated by the example given in Section 4.8.

The initial idea of using exponential data weighting to overcome the numerical problem of a predictive control system was published in Wang (2001b) and Wang (2003).

Problems

- 4.1.** A second-order discrete-time system with time delay is described by the z-transfer function

$$G(z) = \frac{0.1(z - 0.3)}{(z - 1.25)(z - 0.6)} z^{-6}. \quad (4.84)$$

Design a discrete-time predictive control system that will include integral action. The design parameters are specified as $a = 0.5$, $N = 5$, $Q = C^T C$, $R = 1$, however, with the prediction horizon N_p and the weight R as tuning parameters. Compute the matrices Ω and Ψ in the cost function J , where J is defined as

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(k_i).$$

Show that as the prediction horizon N_p increases, the elements in Ω , Ψ and the condition number of Ω increase.

4.2. Continue from Problem 4.1 with further investigation of the effects of the numerical problem when N_p is large.

1. Show that without constraints, the feedback control gain matrix $K_{mpc} = L(0)^T \Omega^{-1} \Psi$ is a function of N_p , also for a large $N_p = 100$, the gain K_{mpc} is numerically sensitive.
2. Suppose that for a unit step set-point change, the limit of the operation only allows $-0.2 \leq \Delta u(k) \leq 0.2$. For this set of simple constraints ($M\eta \leq \gamma$), with $N_p = 100$, investigate how the numerical ill-conditioning problem is carried over to the optimization problem of finding the active constraints. (Hint: from (2.54) in Chapter 2, you can calculate the components in $H = M^T \Omega^{-1} M$ and $K = \gamma + M^T \Omega^{-1} \Psi x(k_i)$ in functions of N_p , and go through Hildreth's programming procedure with an assumed value for $x(k_i)$.)

4.3. The simple modification of discrete-time MPC systems through (4.10) to (4.12) does not guarantee closed-loop stability for all $\alpha > 1$. However, if the original plant is stable with open-loop poles within the unit circle and the unstable modes in the design model come from the augmented integrator(s), the simple approach proposed in (4.10) to (4.12) could produce a stable closed-loop system, provided that the weight α is slightly larger than 1. Good choices are often $\alpha = 1.05$ to 1.1. This could be practical for process control applications where the majority of the plants are stable systems. With this small and simple modification, as the prediction horizon N_p increases, Ω and Ψ will converge. Hence, the numerically ill-conditioning problem is avoided. Test this simple modification with the food extruder process given in Problem 3.8 from Chapter 3, where

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad (4.85)$$

with

$$\begin{aligned} G_{11} &= \frac{0.21048s + 0.00245}{s^3 + 0.302902s^2 + 0.066775s + 0.002186} \\ G_{12} &= \frac{-0.001313s^2 + 0.000548s - 0.000052}{s^4 + 0.210391s^3 + 0.105228s^2 + 0.00777s + 0.000854} \\ G_{21} &= \frac{0.000976s - 0.000226}{s^3 + 0.422036s^2 + 0.091833s + 0.003434} \\ G_{22} &= \frac{-0.00017}{s^2 + 0.060324s + 0.006836}, \end{aligned}$$

and the other conditions remain the same except the use of α weight and change of the prediction horizon as stated as follow. Choose $\alpha = 1.1$, calculate the MPC gain matrix K_{mpc} and the closed-loop eigenvalues as functions of prediction horizon N_p . Increase α until one of the eigenvalues of the closed-loop predictive control system reaches the unit circle on the complex plane.

What is the maximum value of α that is permitted in this application to produce a stable closed-loop system, and what is the condition number of the Hessian matrix Ω ?

4.4. Consider the augmented state-space model

$$\begin{aligned} \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} &= \begin{bmatrix} 0.5 & 0 \\ 0.5 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \Delta u(k) \\ y(k) &= [0 \ 1] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}. \end{aligned}$$

Choose $Q = C^T C$ and $R = 0.1$. Compare the solution of LQR with the solution of predictive control with exponential data weighting.

1. Find the control gain matrix K , Riccati solution P_∞ and the closed-loop eigenvalues of the LQR system using the MATLAB function `dlqr`.
2. With exponential weight $\alpha = 1.5$, Q_α , R_α , $a = 0.3$, $N = 4$ and the prediction horizon of $N_p = 20$, calculate the predictive control gain matrix K_{mpc} and the closed-loop eigenvalues. Q_α and R_α are calculated according to

$$\begin{aligned} \gamma &= \frac{1}{\alpha} \\ Q_\alpha &= \gamma^2 Q + (1 - \gamma^2) P_\infty \\ R_\alpha &= \gamma^2 R. \end{aligned}$$

Compare the results with those from the LQR design. If they are not sufficiently close to the accuracy as you would like, increase the parameter N until you are satisfied.

3. With the initial condition $x(0) = [1 \ 2]^T$, simulate the closed-loop response of the LQR system with K . Separately, for $m = 0, 1, 2, \dots, 30$, simulate the predictive control system with $\Delta \hat{u}(m) = L(m)^T \eta$, where $\eta = -\Omega^{-1} \Psi x(0)$. Scale the responses $\Delta u(m)$ and $x(m)$ from the LQR system by a factor of α^{-m} and compare them with the responses $\Delta \hat{u}(m)$ and $\hat{x}(m)$ from the predictive control system. What are your observations?

4.5. A discrete-time system with two-input and two-output is described by the transfer function model

$$G(z) = \begin{bmatrix} G_{11}(z) & G_{12}(z) \\ G_{21}(z) & G_{22}(z) \end{bmatrix},$$

where the four transfer functions are

$$\begin{aligned} G_{11}(z) &= \frac{0.1(z - 0.5)}{(z - 0.7 + j0.8)(z - 0.7 - j0.8)}, \quad G_{12}(z) = \frac{0.001}{z - 0.3}, \\ G_{21} &= \frac{-0.1}{z^2}, \quad G_{22}(z) = \frac{1.1(z - 0.8)}{(z - 1)^2}. \end{aligned}$$

Design a predictive control system with integral action, which also has the closed-loop poles inside a circle of radius 0.8. The rest of the design specifications are $a_1 = a_2 = 0.5$, $N_1 = N_2 = 12$, $Q = C^T C$ and $R = 0.3$, $\alpha = 1.2$, $N_p = 60$. An observer is needed in the implementation of the predictive control system, where it is designed using the MATLAB `dlqr` function with $Q_{ob} = I$ and $R_{ob} = 0.1I$. Show that through comparison, placing the closed-loop poles inside the circle of radius 0.8 has moved the pair of resonant poles from $0.5779 \pm j0.6732$ to $0.3902 \pm j0.4496$, hence, the closed-loop performance has improved significantly.

When a unit set-point change is applied to y_1 ($r_1(k) = 1$), while $r_2(k) = 0$, assuming zero steady-state and initial conditions, simulate the closed-loop system without constraints. The next step is to impose constraints on the input variables, where they have the following values:

$$\begin{aligned}-1 &\leq \Delta u_1(k) \leq 1, \quad -1 \leq \Delta u_2(k) \leq 1, \\ 8 &\leq u_1(k) \leq 16, \quad -2 \leq u_2(k) \leq 2.\end{aligned}$$

Repeat the simulation with the operating constraints.

4.6. Consider the polymerization process in Tutorial 3.3 (Chapter 3). Utilize the MATLAB program in that tutorial to produce a discrete-time state-space model for this five-input and four-output system. Design a discrete-time MPC with integral action. The closed-loop poles of the MPC system are inside a circle of radius 0.8. The weight matrices are $Q = C^T C$, $R = I$. The exponential weight factor is $\alpha = 1.2$, and the prediction horizon is $N_p = 40$. The parameters in the Laguerre functions, $\alpha_1, \dots, \alpha_5$, and N_1, \dots, N_5 are used as tuning parameters for this predictive control system. Experiment with these tuning parameters and find a combination to achieve a satisfactory response.

Continuous-time Orthonormal Basis Functions

5.1 Introduction

This chapter introduces orthonormal basis functions and their applications in dynamic system modelling. The chapter begins with an introduction to the basic concepts in approximating an arbitrary function with a set of orthonormal basis functions. Laguerre functions not only satisfy these properties, but also possess simple Laplace transforms. The chapter discusses how Laguerre functions are used in modelling the impulse response of a stable system with convergence. A more general class of orthonormal basis functions, called Kautz functions, are introduced towards the end of this chapter. Kautz functions allow complex poles to be used in their structures, however, they require more effort in computing the realization. The modelling idea using a set of orthonormal basis functions forms a fundamental principle of the model predictive control design methods presented in this book. It is helpful if we understand these basic ideas.

5.2 Orthonormal Expansion

A sequence of real functions $l_i(t)$, $i = 1, 2, \dots$ is said to form an orthonormal set over the interval $[0, \infty)$ if they have the property that

$$\int_0^\infty l_i^2(t)dt = 1 \quad (5.1)$$

and

$$\int_0^\infty l_i(t)l_j(t)dt = 0 \quad i \neq j. \quad (5.2)$$

A set of orthonormal functions $l_i(t)$ is said to be complete if the relation

$$\int_0^\infty f(t)l_i(t)dt = 0 \quad (5.3)$$

can only hold for all values of i if the squared $f(t)$ satisfies

$$\int_0^\infty f(t)^2 dt = 0. \quad (5.4)$$

It is known that with respect to a set of functions $l_i(t)$, $i = 1, 2, \dots$ which are orthonormal and complete over the interval $[0, \infty)$, an arbitrary function $f(t)$ has a formal expansion analogous to a Fourier expansion (Wylie, 1960, Lee, 1960). Such an expansion has been widely used in numerical analysis for the approximation of functions in differential and integral equations. In the context of approximation, the function $f(t)$ is written as

$$f(t) = \sum_{i=1}^{\infty} c_i l_i(t), \quad (5.5)$$

where c_i , $i = 1, 2, \dots$ are the coefficients of the expansion and are defined by

$$\begin{aligned} c_1 &= \int_0^\infty l_1(t) f(t) dt \\ c_2 &= \int_0^\infty l_2(t) f(t) dt \\ &\vdots = \vdots \\ c_i &= \int_0^\infty l_i(t) f(t) dt. \end{aligned} \quad (5.6)$$

The expansion (5.5), in theory, has an infinite number of coefficients. However, the assumed completeness of the set of orthonormal functions (Wylie, 1960, Lee, 1960) ensures that for any piece-wise continuous function $f(t)$ with

$$\int_0^\infty f^2(t) dt < \infty, \quad (5.7)$$

and any $\varepsilon > 0$, there exists an integer N such that

$$\int_0^\infty (f(t) - \sum_{i=1}^N c_i l_i(t))^2 dt < \varepsilon. \quad (5.8)$$

Thus, we can approximate the function $f(t)$ arbitrarily closely by $\sum_{i=1}^N c_i l_i(t)$ with an increasing number of terms, N .

5.3 Laguerre Functions

Definition of Laguerre Functions

The Laguerre functions are one set of the orthonormal functions (Lee, 1960) that satisfy the orthonormal and complete properties defined by (5.1) to (5.4). The set of Laguerre functions is defined as, for any $p > 0$,

$$\begin{aligned}
l_1(t) &= \sqrt{2p} \times e^{-pt} \\
l_2(t) &= \sqrt{2p}(-2pt + 1) e^{-pt} \\
&\vdots = \vdots \\
l_i(t) &= \sqrt{2p} \frac{e^{pt}}{(i-1)!} \frac{d^{i-1}}{dt^{i-1}} [t^{i-1} e^{-2pt}].
\end{aligned} \tag{5.9}$$

In the literature, parameter p here is called the time scaling factor for the Laguerre functions. This time scaling factor plays an important role in the application of Laguerre functions, which determines their exponential decay rate. It is used as a design parameter that the user will specify as part of the design requirement.

The Laplace transform of the Laguerre functions (5.9) forms the Laguerre network $L_i(s)$, $i = 1, 2, \dots$, where

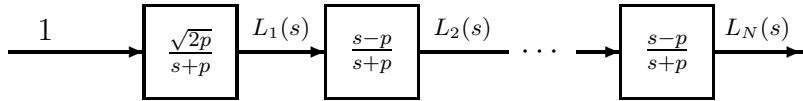
$$\begin{aligned}
L_1(s) &= \int_0^\infty l_1(t)e^{-st} dt = \frac{\sqrt{2p}}{(s+p)} \\
L_2(s) &= \int_0^\infty l_2(t)e^{-st} dt = \frac{\sqrt{2p}(s-p)}{(s+p)^2} \\
&\vdots = \vdots \\
L_i(s) &= \int_0^\infty l_i(t)e^{-st} dt = \frac{\sqrt{2p}(s-p)^{i-1}}{(s+p)^i}.
\end{aligned} \tag{5.10}$$

Each $L_i(s)$ may also be called a Laguerre filter. As we can see from (5.10), the Laguerre filters have simple analytical expressions where all the poles are on the same location p and all filters other than the first one have a first-order filter in series with all-pass filters.¹ Figure 5.1 shows the structure of the N th Laguerre network.

Generating Laguerre Functions

Although Laguerre functions can be generated using (5.9), there is a systematic way to generate the Laguerre functions through the Laguerre networks (5.10). We can derive the Laguerre functions by constructing the state-space form as follows. Define the state vector $L(t) = [l_1(t) \ l_2(t) \dots \ l_N(t)]^T$. Assuming initial conditions of the state vector as $L(0) = \sqrt{2p} [1 \ 1 \ \dots \ 1]^T$, then the Laguerre functions satisfy the state-space equation:

¹ Because the zeros mirror locations of their poles on the complex plane, the amplitude of the frequency response of the filter $\frac{(s-p)^{i-1}}{(s+p)^{i-1}}$ is unity for all frequencies, so it is termed all-pass.

**Fig. 5.1.** Laguerre network

$$\begin{bmatrix} \dot{l}_1(t) \\ \dot{l}_2(t) \\ \vdots \\ \dot{l}_N(t) \end{bmatrix} = \begin{bmatrix} -p & 0 & \dots & 0 \\ -2p & -p & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ -2p & \dots & -2p & -p \end{bmatrix} \begin{bmatrix} l_1(t) \\ l_2(t) \\ \vdots \\ l_N(t) \end{bmatrix}. \quad (5.11)$$

The solution of the differential equation (5.11) gives the set of Laguerre functions for $i = 1, 2, \dots, N$ as

$$L(t) = e^{A_p t} L(0), \quad (5.12)$$

where

$$A_p = \begin{bmatrix} -p & 0 & \dots & 0 \\ -2p & -p & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ -2p & \dots & -2p & -p \end{bmatrix}.$$

This compact state-space representation of Laguerre functions is paramount in the design of continuous-time predictive control. Also note that the matrix A_p is a lower triangular matrix, leading to simplified solutions when used in predictive control.

Tutorial 5.1. Since Laguerre functions are used numerous times in this book, this tutorial demonstrates how to generate these functions effectively and understand how the scaling factor affects the response time of a set of Laguerre functions.

Step by Step

1. Create a new MATLAB function ‘lagic.m’. The input parameters to this function are p and N , and the output parameters from this function are A_p and $L(0)$. Enter the following program into the file:

```

function [Ap,L0]=lagc(p,N)
%Generating system matrix Ap
Ap=-p*eye(N,N);
for ii=1:N
for jj=1:N
if jj<ii, Ap(ii,jj)=-2*p;
end
end
end
L0=sqrt(2*p)*ones(N,1);

```

2. Test this program by creating a new program ‘test.m’. The Laguerre functions are generated via the solution of the differential equation,

$$L(t) = e^{A_p t} L(0).$$

Enter the following program into ‘test.m’.

```

p=1;
N=3;
[Ap,L0]=lagc(p,N);
delta_t=0.01;
Tm=8;
N_sample=Tm/delta_t;
t=0:delta_t:(N_sample-1)*delta_t;
%solution of the differential equation
for i=1:N_sample;
L(:,i)=expm(Ap*t(i))*L0;
end
figure
plot(t,L(1,:),t,L(2,:),'--',t,L(3,:),'.')
xlabel('Time (sec)')

```

3. Run ‘test.m’. Figure 5.2 shows the first three Laguerre functions where the scaling factor $p = 1$.
4. Select $p = 0.1$ and $p = 10$, and compare the Laguerre functions for $i = 1, 2, 3$. What is your observation of the effect of the time scaling factor p on the Laguerre functions?

5.4 Approximating Impulse Responses

This set of orthonormal basis functions has been widely used in mathematical modelling and system identification. The idea behind system approximation using Laguerre functions is to take the impulse response of the system to be

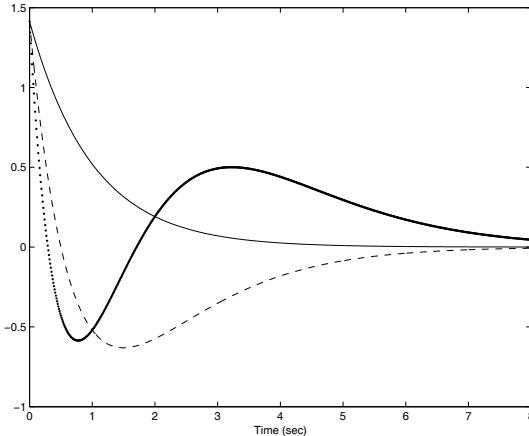


Fig. 5.2. Laguerre functions ($p = 1$). Solid line: $l_1(t)$; dashed line: $l_2(t)$; dotted line: $l_3(t)$

$h(t)$ (Lee, 1960). We write the impulse response $h(t)$ in terms of an orthonormal expansion using Laguerre functions. That is

$$h(t) = c_1 l_1(t) + c_2 l_2(t) + \dots + c_i l_i(t) + \dots \quad (5.13)$$

The expansion (5.13), in theory, requires an infinite number of terms, unless the impulse response function $h(t)$ satisfies the following condition:

$$\int_0^\infty h^2(t) dt < \infty. \quad (5.14)$$

Equation (5.14) is the condition for a system being L^2 stable. A system that satisfies the L^2 stability condition will have all poles strictly on the left-half complex plane, and its Laplace transfer function is strictly proper. For this class of systems, for any $\varepsilon > 0$, there exists an integer N such that

$$\int_0^\infty (h(t) - \sum_{i=1}^N c_i l_i(t))^2 dt < \varepsilon. \quad (5.15)$$

Thus, we can approximate the function $h(t)$ arbitrarily closely by $\sum_{i=1}^N c_i l_i(t)$ with an increasing number of terms, N . This is true for an arbitrary choice of scaling factor $p > 0$. This basically says that for a given function $h(t)$ satisfying (5.14), with a choice of $p > 0$, the accuracy of the approximation using an orthonormal expansion increases as the number of terms, N , increases. Namely the expansion converges to the function it tries to model.

If the function $h(t)$ is given, the optimal coefficients c_i , $i = 1, 2, \dots, N$ can be found by minimizing the cost function:

$$J = \int_0^\infty (h(t) - \sum_{i=1}^N c_i l_i(t))^2 dt. \quad (5.16)$$

Noting the orthonormal property of the Laguerre functions, the optimal solutions are given by

$$\begin{aligned} c_1 &= \int_0^\infty l_1(t) h(t) dt \\ c_2 &= \int_0^\infty l_2(t) h(t) dt \\ &\vdots \\ c_N &= \int_0^\infty l_N(t) h(t) dt. \end{aligned} \quad (5.17)$$

In other applications, if the impulse response is not explicitly given, alternative means and criteria need to be found to optimally determine the coefficients c_i , $i = 1, 2, 3, \dots$. This is exactly what we do in the design of model predictive control using Laguerre functions.

Note that the coefficients are optimal with respect to a pre-determined Laguerre scaling factor p . For a finite N and given $p > 0$, the integral squared error between the function $h(t)$ and the impulse response of the Laguerre model is defined and calculated by

$$\int_0^\infty (h(t) - \sum_{i=1}^N c_i l_i(t))^2 dt = \sum_{i=N+1}^\infty c_i^2, \quad (5.18)$$

where we used the orthonormal properties and the definition of the optimal coefficients to obtain the result. For a given number of terms N , the scaling factor p could be optimized with respect to the integral squared error by maximizing $\sum_{i=1}^N c_i^2$ (Wang and Cluett, 2000).

To make us familiar with the approximation of the impulse response of a stable system, we introduce a tutorial as follows.

Tutorial 5.2. *In this tutorial, we find the approximate Laguerre model based on the impulse response of a system, $h(t)$, where the impulse is generated based on the Laplace transfer function $G(s) = \frac{1}{(s+1)(s+2)}$. The scaling factor $p = 1$ and $N = 4$ are used in the simulation. The sampling interval $\Delta t = 0.01$.*

We first introduce a simple numerical integration function based on Simpson's rule, which will be used to calculate the Laguerre coefficients.

Step by Step

1. Create a function called 'inte.m'. Enter the following program into the file:

```
function v=inte(F,h)
```

```
%Use Simpson's Rule to compute an integral expression
% F is the discretized function
% h is the sampling interval
% v is the computed value
[m,n]=size(F);
if m==1 m=n;
end
if n==1 n=m; end
NN=(n-1)/2;
s=0;
for j=2:2:2*NN
s=s+4*F(j);
end
for j=3:2:2*NN-1
s=s+2*F(j);
end
v=(s+F(1)+F(2*NN+1))*h/3;
```

2. This simple function is useful for computing numerical integrations, which provides better accuracy than other simple schemes.
3. We next create a new function called 'lagmodel.m'. The four Laguerre functions will be generated first. Enter the following program into the file (lagmodel.m):

```
p=1;
N=4;
delta_t=0.01;
Tm=8;
N_sample=Tm/delta_t;
t=0:delta_t:(N_sample-1)*delta_t;
[Ap,L0]=lagc(p,N);
for i=1:N_sample;
L(:,i)=expm(Ap*t(i))*L0;
end
```

4. Generate the impulse response of the system. Continue entering the following program into the file:

```
num=1;
den=conv([1 1],[1 2]);
h_t=impulse(num,den,t);
```

5. Solve the integral equations using Simpson's rule to get the four Laguerre coefficients. Continue entering the following program into the file:

```
hL_1=L(1,:).*h_t;
c1=INTE(hL_1,delta_t);
hL_2=L(2,:).*h_t;
```

```
c2=INTE(hL_2,delta_t);
hL_3=L(3,:).*h_t;
c3=INTE(hL_3,delta_t);
hL_4=L(4,:).*h_t;
c4=INTE(hL_4,delta_t);
```

6. We will construct two models for comparative study. One is a second-order model and one is a fourth-order model. Continue entering the following program into the file:

```
h_model2=c1*L(1,:)+c2*L(2,:);
h_model4=c1*L(1,:)+c2*L(2,:)+c3*L(3,:)+c4*L(4,:);
figure subplot(211)
plot(t,h_t,t,h_model2,'--');
xlabel('Time(sec)')
subplot(212)
plot(t,h_t,t,h_model4,'--');
xlabel('Time (sec)')
```

7. Run ‘lagmodel.m’. Figure 5.3 shows the impulse response of the Laguerre model in comparison to the true response. By comparing the top plot with the bottom plot, it is seen that as the number of terms N increases from 2 to 4, the accuracy of the approximation increases.

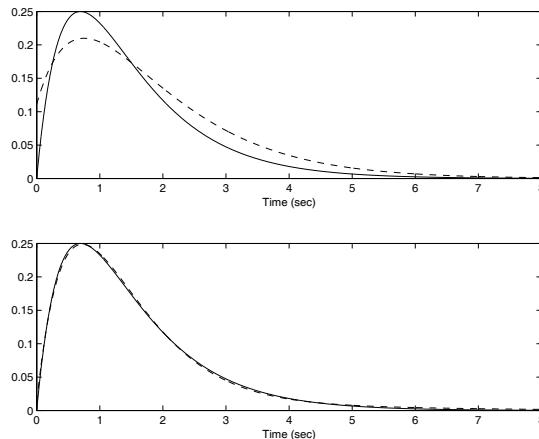


Fig. 5.3. Approximate impulse response. Top plot: solid-line, true impulse response; dashed line second-order Laguerre model response. Bottom plot: solid-line, true impulse response; dashed line, fourth-order Laguerre model response

5.5 Kautz Functions

Perhaps one of the reasons why Laguerre functions have been so popular in system identification and control is their simplicity. The properties of the functions are predominately determined by the scaling factor p , and the complexity of the functions increases as the order N increases (see Figure 5.2). However, this simplicity can also lead to limitations in the application of Laguerre networks, particularly in the situation where the underlying system has complex poles. The Kautz networks discussed in the following will overcome the drawback of Laguerre networks by allowing non-identical poles to be used in the construction of the networks, as well as complex poles.

Kautz Networks

Kautz networks were first proposed by Kautz in his 1954 paper (Kautz, 1954). The original Kautz networks were used for modelling an impulse response of a continuous-time system, as a generalization of the original work by Wiener and Lee in the 1930s (see Lee, 1960). At that time, there were no computational tools to obtain solutions and only analytical solutions were available, hence the Kautz networks were derived through orthonormality analysis using the location of poles and zeros of the networks (Lee, 1960, Kautz, 1954). Kautz functions in the time domain were derived through the inverse Laplace transform.

There are three basic cases in the construction of Kautz networks. They offer different degrees of complexity.

Case A. Kautz networks with non-identical real poles

Let $-p_1, -p_2, \dots, -p_N$ be the real pole locations for the Kautz networks. For all $p_k > 0$, the Kautz networks for $k = 1, 2, \dots, N$ were proposed as

$$\begin{aligned} K_1(s) &= \sqrt{2p_1} \frac{1}{s + p_1} \\ K_2(s) &= \sqrt{2p_2} \frac{1}{s + p_2} \frac{s - p_1}{s + p_1} \\ K_3(s) &= \sqrt{2p_3} \frac{1}{s + p_3} \frac{s - p_2}{s + p_2} \frac{s - p_1}{s + p_1} \\ &\vdots \\ K_N(s) &= \sqrt{2p_N} \frac{1}{s + p_N} \prod_{i=1}^{N-1} \frac{s - p_i}{s + p_i}. \end{aligned}$$

This case is the closest one to the Laguerre network. When all the poles are chosen to be identical, this Kautz network becomes a Laguerre network. Figure 5.4 shows the block structure of this Kautz network.

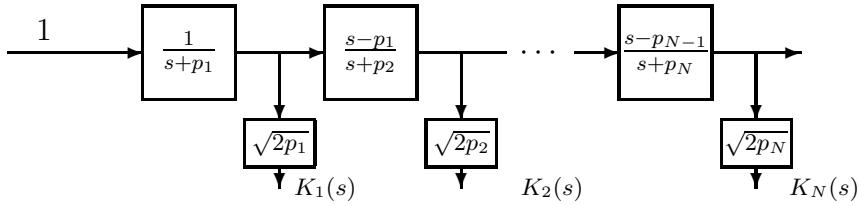


Fig. 5.4. Kautz network with real poles

Case B. Kautz networks with non-identical complex poles

Let $n = \frac{N}{2}$, where N is the number of terms in the networks. Denote the complex poles as $-\alpha_i \pm j\beta_i$ for $i = 1, 2, \dots, n$ where $\alpha_i > 0$ for all i . We also denote $\gamma_i = \sqrt{\alpha_i^2 + \beta_i^2}$ as the parameters used in the parametrization of the Kautz networks. The Kautz networks with complex poles are defined as

$$\begin{aligned}
 K_1(s) &= \sqrt{2\alpha_1} \frac{s + \gamma_1}{(s + \alpha_1 + j\beta_1)(s + \alpha_1 - j\beta_1)} \\
 K_2(s) &= \sqrt{2\alpha_1} \frac{s - \gamma_1}{(s + \alpha_1 + j\beta_1)(s + \alpha_1 - j\beta_1)} \\
 K_3(s) &= \sqrt{2\alpha_2} \frac{s + \gamma_2}{(s + \alpha_2 + j\beta_2)(s + \alpha_2 - j\beta_2)} \frac{(s - \alpha_1 - j\beta_1)(s - \alpha_1 + j\beta_1)}{(s + \alpha_1 + j\beta_1)(s + \alpha_1 - j\beta_1)} \\
 K_4(s) &= \sqrt{2\alpha_2} \frac{s - \gamma_2}{(s - \alpha_2 + j\beta_2)(s + \alpha_2 - j\beta_2)} \frac{(s - \alpha_1 - j\beta_1)(s - \alpha_1 + j\beta_1)}{(s + \alpha_1 + j\beta_1)(s + \alpha_1 - j\beta_1)} \\
 &\vdots \\
 K_{2n-1}(s) &= \frac{\sqrt{2\alpha_{n/2}}(s + \gamma_{n/2})}{(s + \alpha_{n/2} + j\beta_{n/2})(s + \alpha_{n/2} - j\beta_{n/2})} \Gamma_n(s) \\
 K_{2n}(s) &= \frac{\sqrt{2\alpha_{n/2}}(s - \gamma_{n/2})}{(s + \alpha_{n/2} + j\beta_{n/2})(s + \alpha_{n/2} - j\beta_{n/2})} \Gamma_n(s),
 \end{aligned}$$

where $\Gamma_n(s)$ is defined by the following network:

$$\Gamma_n(s) = \prod_{i=1}^{n/2-2} \frac{(s - \alpha_i - j\beta_i)(s - \alpha_i + j\beta_i)}{(s + \alpha_i + j\beta_i)(s + \alpha_i - j\beta_i)}.$$

Figure 5.5 shows the Kautz network with two pairs of complex poles. A simplified representation of the networks is to use only one pair of complex poles by letting $\alpha_1 = \alpha_2 = \dots = \alpha$, $\beta_1 = \beta_2 = \dots = \beta$, $\gamma_1 = \gamma_2 = \dots = \gamma$. This representation becomes similar to the Laguerre networks.

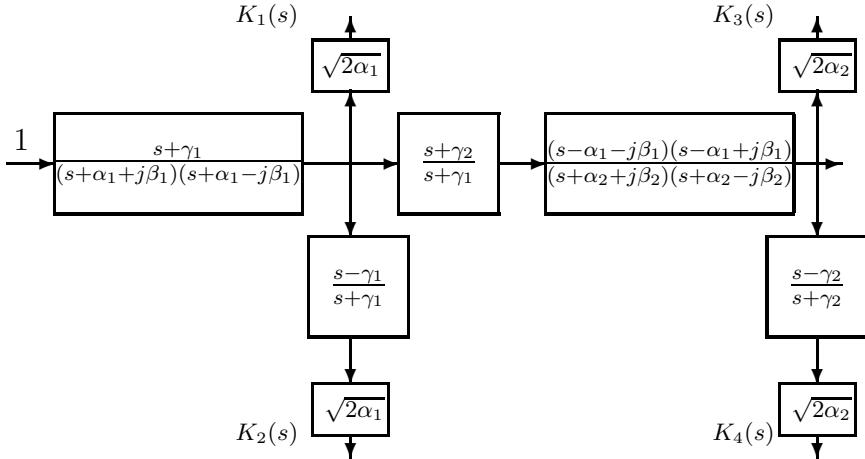


Fig. 5.5. Kautz network with two pairs of complex poles, $\gamma_1 = \sqrt{\alpha_1^2 + \beta_1^2}$, $\gamma_2 = \sqrt{\alpha_2^2 + \beta_2^2}$

Case C. Combination of real and complex poles in Kautz network

When putting together a Kautz network with real poles and one with complex poles, it is best to first construct the network for the part with all the real poles, followed by the part with all the complex poles. To illustrate this idea, we assume a Kautz network having a real pole at $-p_1$ and a pair of complex poles at $-\alpha_1 \pm j\beta_1$. Then, the Kautz network is constructed as below:

$$\begin{aligned} K_1(s) &= \frac{\sqrt{2p_1}}{s + p_1} \\ K_2(s) &= \sqrt{2\alpha_1} \frac{s + \gamma_1}{(s + \alpha_1 + j\beta_1)(s + \alpha_1 - j\beta_1)} \frac{(s - p_1)}{(s + p_1)} \\ K_3(s) &= \sqrt{2\alpha_1} \frac{s - \gamma_1}{(s + \alpha_1 + j\beta_1)(s + \alpha_1 - j\beta_1)} \frac{(s - p_1)}{(s + p_1)}. \end{aligned}$$

5.5.1 Kautz Functions in the Time Domain

All Kautz functions, both real and complex, are included in one network, where the dimension of the Kautz network $N = n_{real} + 3 + 4(n_{complex} - 1)$. Let $F(t)$ denotes the state vector for the Kautz functions. Then, we express the Kautz functions using the state-space model:

$$\begin{aligned} \dot{F}(t) &= A_k F(t) + B_k \delta(t) \\ K(t) &= C_k F(t), \end{aligned} \tag{5.19}$$

where we assume zero initial condition for $F(t)$ and $\delta(t)$ is the unit impulse function. The matrices A_k , B_k and C_k are determined by the locations of the poles of the Kautz network. However, the realization is chosen such that the system matrix A_k is a lower triangular matrix which is important in the applications of Kautz functions for design of continuous-time predictive control systems (See Chapter 6) to ensure closed-form solutions.

Example 5.1. Suppose that a Kautz network has three real poles $-\alpha_1$, $-\alpha_2$, and $-\alpha_3$, where $\alpha_1, \alpha_2, \alpha_3 > 0$. Find the state-space representation of the Kautz functions.

Solution. By choosing $F(t) = [f_1(t) f_2(t) f_3(t)]^T$, the Kautz functions $K(t) = [k_1(t) k_2(t) k_3(t)]^T$ are described by the differential equation (5.19) with A_k , B_k and C_k given by

$$A_k = \begin{bmatrix} -\alpha_1 & 0 & 0 \\ -2\alpha_1 & -\alpha_2 & 0 \\ -2\alpha_1 & -\alpha_2 & -\alpha_3 \end{bmatrix}; B_k = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}; C_k = \begin{bmatrix} \sqrt{2\alpha_1} & 0 & 0 \\ 0 & \sqrt{2\alpha_2} & 0 \\ 0 & 0 & \sqrt{2\alpha_3} \end{bmatrix}.$$

Example 5.2. Suppose that a Kautz network has two poles at $-\alpha_1$ and $-\alpha_2$, and a pair of complex poles at $-\alpha_3 \pm j\beta_3$. Find the state-space representation of the Kautz functions.

Solution. We define $p_3 = -\alpha_3 + j\beta_3$ and $\bar{p}_3 = -\alpha_3 - j\beta_3$. $\alpha_1, \alpha_2, \alpha_3 > 0$. With $\gamma_3 = \sqrt{\alpha_3^2 + \beta_3^2}$, the Kautz functions $K(t) = [k_1(t) k_2(t) k_3(t) k_4(t)]^T$ are described by the differential equation (5.19) with

$$A_k = \begin{bmatrix} -\alpha_1 & 0 & 0 & 0 & 0 \\ -2\alpha_1 & -\alpha_2 & 0 & 0 & 0 \\ -2\alpha_1 & -\alpha_2 & -p_3 & 0 & 0 \\ -2\alpha_1 & -\alpha_2 & \gamma_3 - p_3 & -\bar{p}_3 & 0 \\ -2\alpha_1 & -\alpha_2 & \gamma_3 - p_3 & -\gamma_3 - \bar{p}_3 & -\gamma_3 \end{bmatrix}; B_k = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$C_k = \begin{bmatrix} \sqrt{2\alpha_1} & 0 & 0 & 0 & 0 \\ 0 & \sqrt{2\alpha_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{2\alpha_3} & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2\alpha_3} \end{bmatrix}.$$

5.5.2 Modelling the System Impulse Response

Similar to the Laguerre model, the approximate model using a Kautz network is given by

$$h(t) = \sum_{i=1}^{\infty} c_i k_i(t), \quad (5.20)$$

where the coefficients are defined by

$$\begin{aligned}
c_1 &= \int_0^\infty k_1(t)h(t)dt \\
c_2 &= \int_0^\infty k_2(t)h(t)dt \\
&\vdots = \vdots \\
c_i &= \int_0^\infty k_i(t)h(t)dt.
\end{aligned} \tag{5.21}$$

The integral squared error for the Kautz model with order N is defined by

$$\int_0^\infty (h(t) - \sum_{i=1}^N c_i k_i(t))^2 dt = \sum_{i=N+1}^{\infty} c_i^2. \tag{5.22}$$

For a given transfer function, the integral squared error can be made to be zero if we choose N to be equal to the model order and the set of poles in the Kautz filters to be equal to the poles of the system. This means, in theory, that by correctly choosing the poles of the Kautz filters, the Kautz model can be made to be identical to the underlying system. This is illustrated by the following numerical example.

Example 5.3. Suppose that the transfer function of a continuous-time system is

$$G(s) = \frac{(-s + 1)}{(s + 1.1)(s + 0.7)(s + 0.1)}.$$

Choose the three poles in the Kautz network to be equal to the system's poles, namely $p_1 = 1.1, p_2 = 0.7, p_3 = 0.1$, and find the Kautz model for this system. Evaluate the integral squared error.

Solution. We first generate the impulse response of this system by using sampling interval $\Delta t = 0.01$ (sec) and response time 48 (sec). We also use the same set of parameters to generate the Kautz functions via the solution of differential equation (5.19). Then, we compute the coefficients of the Kautz network to obtain $c_1 = -0.0312, c_2 = -0.9293, c_3 = 2.5572$ from which we construct the Kautz model as

$$h_k(t) = c_1 k_1(t) + c_2 k_2(t) + c_3 k_3(t).$$

The integral squared error between the impulse response of the system and the Kautz model is 4.4307×10^{-7} .

5.6 Summary

This chapter has introduced orthonormal basis functions. There are two main classes of orthonormal basis functions that have been widely used in the control and systems community. One is the set of Laguerre functions; and the

other is the set of Kautz functions. Both sets of functions have been used in the design of predictive control systems.

The important assumption when applying the Laguerre functions or the Kautz functions is that the integral squared value of the time function to be captured by the basis functions is required to be bounded. If this is the case, then the accuracy of the approximation increases as the number of terms increases. Namely convergence is guaranteed.

Problems

5.1. Based on Tutorial 5.2 and the plant transfer function $G(s) = \frac{1}{(s+1)(s+2)}$, identify a Laguerre model with the number of terms $N = 3$, where the optimal scaling factor p is found by maximizing $\sum_{i=1}^3 c_i^2$.

5.2. Continuous-time Laguerre models are useful to capture the process dynamics from plant impulse or step response data (Wang and Cluett, 2000).

Consider the 12th- order continuous-time system described by the transfer function

$$G(s) = \frac{(15s + 1)^2(4s + 1)(2s + 1)}{(20s + 1)^3(10s + 1)^3(5s + 1)^3(0.5s + 1)^3}.$$

This higher-order system was approximated quite accurately using a Laguerre model with 6 terms and $p = 0.063$. Following Tutorial 5.2, find the approximate Laguerre transfer function model for this higher-order system. Verify that the equivalent state-space model is (A_m, B_m, C_m) , where A_m equals A_p and B_m equals $L(0)$ and $C_m = [c_1 \ c_2 \ \dots \ c_6]$.

5.3. Suppose that we have a severely underdamped system with transfer function $G(s) = \frac{1}{(s+1+j6)(s+1-j6)}$. If we approximate this system with a Laguerre network, we will observe a large error for a lower-order Laguerre model. However, the error reduces as the number of terms N increases. From your observation, how many terms would be sufficient to describe this system (assuming $p = 1$)? Repeat the modelling process with Kautz filters by choosing a pair of complex poles that are identical to the system poles. Show that if you use a fourth-order Kautz model with two identical pairs of the complex poles as you first chose, the coefficients for the two additional filters are negligible.

Continuous-time MPC

6.1 Introduction

This chapter discusses continuous-time model predictive control (CMPC) without constraints. It will take the reader through the principles of continuous-time predictive control design, and the solutions of the optimal control problem. It shows that when constraints are not involved in the design, the continuous-time model predictive control scheme becomes a state feedback control system, with the gain being chosen from minimizing a finite prediction horizon cost. The continuous-time Laguerre functions and Kautz functions discussed in Chapter 5 are utilized in the design of continuous-time model predictive control. When a set of Laguerre functions is used in the design, the desired closed-loop response can be achieved by tuning the time scaling factor p and the number of terms N . Without constraints, the model predictive control has an analytical optimal solution. Since constant input disturbance rejection and set-point following are the most commonly encountered design requirements, this chapter will treat those cases extensively by embedding an integrator in the design model.

This chapter concludes with the use of Kautz functions in the design, which leads to explicit specification of the poles in the orthonormal functions. Without constraints, if numerically permitted for a sufficiently large prediction horizon, this could become entirely identical to the underlying continuous-time linear quadratic regulator (LQR).

6.2 Model Structures for CMPC Design

The design principle of a continuous-time predictive control is very similar to the one used in the discrete-time counterpart. More explicitly, the receding horizon control idea is used in the continuous-time design. However, the difference is that the continuous-time design is based on a continuous-time

state-space model, and all the design parameters, such as, the prediction horizon and prediction time are based on actual time, instead of the number of samples as in the discrete-time case. Although the model and design are based on continuous time, the implementation of a continuous-time predictive control system is performed in a digital environment, which may give advantages to systems with a fast sampling rate or an irregular sampling rate.

The general design philosophy of model predictive control is to compute a trajectory of a future manipulated variable $u(t)$ to optimize the future behaviour of the plant output $y(t)$. The optimization is performed within a limited time window. This time-dependent window for optimization is described by an initial time, t_i , and the length of the window, T_p . With a given initial time t_i , the window is taken from t_i to $t_i + T_p$. The length of the window T_p remains constant. Similar to discrete-time MPC, the prediction horizon equals the length of the moving horizon window T_p and dictates how ‘far’ we wish the future to be predicted. Unlike the discrete-time case, there is no explicit control horizon parameter. This is because it is difficult to determine when exactly a control signal represented by a set of exponential functions will reach a steady state before the optimization is performed. Similar to the discrete-time case, based on the receding horizon control principle, although the optimal trajectory of the future control signal is completely captured within the moving horizon window, the actual control input to the plant only takes the first instant of the control signal, while neglecting the rest of the trajectory.

The key to the design of a continuous-time predictive control system is to model the control trajectory using a set of orthonormal functions. In order to achieve convergence with an increase of complexity of the orthonormal model structure, as stated in Chapter 5, the underlying control trajectory is required to satisfy the property that its integral squared value is bounded. Or explicitly, if the underlying control signal is called $u(t)$, then

$$\int_0^\infty u(t)^2 dt < \infty. \quad (6.1)$$

It is a fact that if there is an external constant input signal to the control system, such as a set-point signal or a constant disturbance signal, the control signal itself does not satisfy (6.1). This is because in order to follow the set-point signal, the control signal needs to converge to a non-zero constant that is related to the steady-state gain of the plant and the magnitude of the set-point change. Therefore, instead of modelling the control signal, the continuous-time predictive control design will target the derivative of the control signal, $\dot{u}(t)$, which will satisfy the property,

$$\int_0^\infty \dot{u}(t)^2 dt < \infty, \quad (6.2)$$

for external constant input signals. A by-product of this choice is that the predictive control system will have integral action, which will be shown later.

6.2.1 Model Structure

The objective here is to propose the model structure that will use the derivative of the control as its input signal, while maintaining the same output. As a result, an integrator is embedded into the design model.

Suppose that the plant to be controlled is an m -input and q -output multivariable system having a state-space model

$$\begin{aligned}\dot{x}_m(t) &= A_m x_m(t) + B_m u(t) \\ y(t) &= C_m x_m(t),\end{aligned}\quad (6.3)$$

where $x_m(t)$ is the state vector of dimension n_1 . In (6.3), A_m , B_m and C_m have dimension $n_1 \times n_1$, $n_1 \times m$ and $q \times n_1$, respectively. We assume that the number of outputs is less than or equal to the number of inputs (*i.e.*, $q \leq m$). If the number of outputs is greater than the number of inputs, we cannot hope to control each of the measured outputs independently with zero steady-state errors.

Note that a general formulation of a continuous-time state-space model has a direct term from the input signal $u(t)$ to the output $y(t)$ as

$$y(t) = C_m x_m(t) + D_m u(t).$$

However, due to the principle of receding horizon control, where a current information of the plant is required for prediction and control, we have implicitly assumed that the input $u(t)$ cannot affect the output $y(t)$ at the same time. Thus, $D_m = 0$ in the plant model.

Let us now define the auxiliary variables

$$\begin{aligned}z(t) &= \dot{x}_m(t) \\ y(t) &= C_m x_m(t),\end{aligned}$$

and choose a new state variable vector $x(t) = [z(t)^T \ y(t)^T]^T$. With these auxiliary variables, in conjunction with (6.3), the augmented state-space model is defined as

$$\begin{bmatrix} \dot{z}(t) \\ \dot{y}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} A_m & o_m^T \\ C_m & o_{q \times q} \end{bmatrix}}_A \begin{bmatrix} z(t) \\ y(t) \end{bmatrix} + \underbrace{\begin{bmatrix} B_m \\ o_{q \times m} \end{bmatrix}}_B \dot{u}(t) \quad (6.4)$$

$$y(t) = \underbrace{\begin{bmatrix} o_m & I_{q \times q} \end{bmatrix}}_C \begin{bmatrix} z(t) \\ y(t) \end{bmatrix}, \quad (6.5)$$

where $I_{q \times q}$ is the identity matrix with dimensions $q \times q$; $o_{q \times q}$ is a $q \times q$ zero matrix, $o_{q \times m}$ is a $q \times m$ zero matrix, and o_m is a $q \times n_1$ zero matrix. For notational simplicity, the augmented model (6.5) is denoted by the matrices (A, B, C) , where explicitly

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t),\end{aligned}\tag{6.6}$$

where A , B and C are matrices corresponding to the forms given in (6.5). In the following, the dimensionality of the augmented state-space equation is taken to be n ($= n_1 + q$). Note that the augmented state-space description (6.6) has the first derivative of the control signal as its input and its output remains the same.

In the presence of a disturbance, we assume that the plant model has the form:

$$\begin{aligned}\dot{x}_m(t) &= A_m x_m(t) + B_m u(t) + B_d \omega(t) \\ y(t) &= C_m x_m(t),\end{aligned}\tag{6.7}$$

where $\omega(t)$ represents unmeasured disturbance sources. We will discuss two cases. The first case is the deterministic disturbance, followed by the stochastic disturbance case.

Model for deterministic disturbance

We consider the case of a deterministic disturbance, where $\omega(t)$ is a constant so that

$$\frac{d\omega(t)}{dt} = 0.\tag{6.8}$$

Again, let us now define the auxiliary variables

$$\begin{aligned}z(t) &= \dot{x}_m(t) \\ y(t) &= C_m x_m(t),\end{aligned}$$

and choose a new state variable vector $x(t) = [z(t)^T \ y(t)^T]^T$. With these auxiliary variables, the augmented state space model is identical to the model without disturbance (see (6.5)), where the differentiation of a constant disturbance becomes zero.

Stochastic disturbance

In the case where there is a random disturbance in the system, we assume that the input disturbance is a source of continuous-time integrated white noise. Namely

$$\omega(t) = \int_0^t \epsilon(\tau) d\tau,\tag{6.9}$$

where $\epsilon(\cdot)$ is a band-limited, zero-mean, white noise. With this,

$$E\left\{\frac{d\omega(t)}{dt}\right\} = E\{\epsilon(t)\} = 0,\tag{6.10}$$

and

$$E\{\epsilon(t)\epsilon(\tau)\} = W_\omega \delta(t - \tau),$$

where $E\{\}$ denotes expectation and $\delta(\cdot)$ is the Dirac function. The augmented model with input integrated white noise is expressed as

$$\begin{aligned} \begin{bmatrix} \dot{z}(t) \\ \dot{y}(t) \end{bmatrix} &= \underbrace{\begin{bmatrix} A_m & o_m^T \\ C_m & o_{q \times q} \end{bmatrix}}_A \begin{bmatrix} z(t) \\ y(t) \end{bmatrix} + \underbrace{\begin{bmatrix} B_m \\ o_{q \times m} \end{bmatrix}}_B \dot{u}(t) + \underbrace{\begin{bmatrix} B_d \\ o_{q \times m} \end{bmatrix}}_{B_e} \epsilon(t) \\ y(t) &= \underbrace{\begin{bmatrix} o_m & I_{q \times q} \end{bmatrix}}_C \begin{bmatrix} z(t) \\ y(t) \end{bmatrix}. \end{aligned} \quad (6.11)$$

The key here is that the plant integrated white noise becomes zero-mean, white noise in the augmented model (6.11). Thus, when the augmented model is used for prediction, the effect of disturbance in expectance on the future state variable is zero.

Example 6.1. Assuming that a constant input disturbance is d , find the augmented model for the system

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} d \\ y(t) &= [1 \ 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \end{aligned} \quad (6.12)$$

Show that the augmented model is both controllable and observable. Furthermore, design a state feedback control system with closed-loop poles allocated at $-3\omega_0$ and examine the behaviour of the derivative of the control in the presence of a constant input disturbance.

Solution. Using the formulation defined by (6.4) and (6.5), the augmented model for (6.12) is obtained as below, where the input to the model is the derivative of the control signal and the derivative of the constant disturbance is 0

$$\begin{bmatrix} \ddot{x}_1(t) \\ \ddot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\omega_0^2 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ x_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \dot{u}(t) \quad (6.13)$$

$$y(t) = [0 \ 0 \ 1] \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ x_3(t) \end{bmatrix}. \quad (6.14)$$

The eigenvalues for this system are $\lambda_1 = j\omega_0$, $\lambda_2 = -j\omega_0$ and $\lambda_3 = 0$. The augmented model is both controllable and observable, which can be checked

through the computation of the controllability and observability matrices. Here, the controllability matrix is calculated as

$$[B \ AB \ A^2B] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & -\omega_0^2 \\ 0 & 0 & 1 \end{bmatrix},$$

and the observability matrix is calculated as

$$\begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Both matrices have non-zero determinant. Therefore, the augmented model is both controllable and observable.

Here, the desired closed-loop polynomial is selected as $(s + 3\omega_0)^3$. The feedback control law is defined as

$$\dot{u}(t) = -[k_1 \ k_2 \ k_3] \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix}. \quad (6.15)$$

The closed-loop characteristic polynomial is calculated via

$$\det \begin{bmatrix} s & -1 & 0 \\ \omega_0^2 + k_1 & s + k_2 & k_3 \\ -1 & 0 & s \end{bmatrix} = s^3 + k_2 s^2 + (\omega_0^2 + k_1)s + k_3.$$

By equating the closed-loop characteristic polynomial to the desired closed-loop polynomial, we find the coefficients for the controller as $k_1 = 26\omega_0^2$, $k_2 = 9\omega_0$ and $k_3 = 27\omega_0^3$.

To examine the behaviour of the closed-loop system, we note that the derivative of the constant input disturbance is zero, *i.e.*, $\dot{d}(t) = 0$. The closed-loop system is

$$\begin{bmatrix} \ddot{x}_1(t) \\ \ddot{x}_2(t) \\ \ddot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -27\omega_0^2 & -9\omega_0^2 & -27\omega_0^2 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} \quad (6.16)$$

$$\dot{u}(t) = -[k_1 \ k_2 \ k_3] \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix}. \quad (6.17)$$

Figure 6.1a shows that the derivative $\dot{u}(t)$ exponentially decays to zero, and Figure 6.1b shows the area under the plot $\dot{u}(t)^2$ is bounded for an arbitrarily large t . Since the signal $u(t)$ is an exponentially decay function that goes to zero, $\int_0^\infty \dot{u}(t)^2 dt < \infty$. In conjunction with the discussion given in Sections 5.2 and 5.3, this example demonstrated that the derivative of the control signal is a good candidate to be modelled by using a set of Laguerre functions.

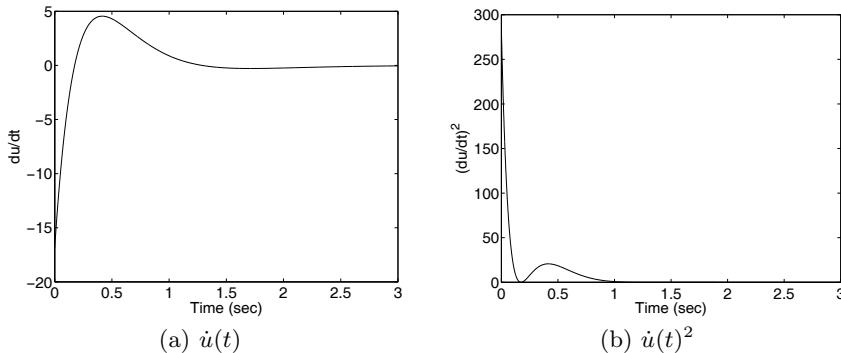


Fig. 6.1. Derivative of the control signal

6.2.2 Controllability and Observability of the Model

It is important to investigate the controllability and observability of the augmented model (6.5). Because the original plant model is augmented with integrators and the MPC design is performed on the basis of the augmented continuous-time state-space model, it is important for control system design that the augmented model does not become uncontrollable or unobservable, particularly with respect to the unstable dynamics of the system. Similar to discrete-time systems, controllability is a pre-requisite for the continuous-time predictive control system to achieve desired closed-loop control performance and observability is a pre-requisite for a successful design of an observer, with a desired performance.

The investigation is based on the transfer function of the state-space model (6.6) in relation to the plant model (6.3). The relationship between a state-space system denoted by (A, B, C) and the transfer function $G(s)$ is described by the following definition.

Definition: A realization of transfer function $G(s)$ is any state-space triplet (A, B, C) such that $G(s) = C(sI - A)^{-1}B$. If such a set (A, B, C) exists, then $G(s)$ is said to be realizable. A realization (A, B, C) is called a minimal realization of a transfer function if there is no other realization of smaller state dimension.

A minimal realization has this distinctive feature, summarized in the theorem below.

Theorem 6.1. *A minimal realization is both controllable and observable.*

The proof of this results can be found in (Kailath, 1980, Bay, 1999). With this background information, the aim is to show conditions such that the augmented model is both controllable and observable through the argument of minimal realization.

Theorem 6.2. Assume that the plant model (A_m, B_m, C_m) in (6.3) is both controllable and observable, having the transfer function $G_m(s)$ with minimal realization, where

$$G_m(s) = C_m(sI - A_m)^{-1}B_m.$$

Then the transfer function of augmented design model (6.5) has the representation

$$G(s) = \frac{1}{s}G_m(s), \quad (6.18)$$

and is both controllable and observable if and only if the plant model $G_m(s)$ has no zero at $s = 0$.¹

Proof. To prove that the augmented model is controllable and observable, we need to show that (6.18) is true. After that, the results follow from the minimal structure of the augmented model without pole-zero cancellation. Note that for a given square matrix M with the block structure

$$M = \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix},$$

if A_{11}^{-1} and A_{22}^{-1} exist, then

$$M^{-1} = \begin{bmatrix} A_{11}^{-1} & 0 \\ -A_{22}^{-1}A_{21}A_{11}^{-1} & A_{22}^{-1} \end{bmatrix}. \quad (6.19)$$

By applying the equality (6.19), we obtain

$$G(s) = C(sI - A)^{-1}B \quad (6.20)$$

where

$$(sI - A)^{-1} = \begin{bmatrix} (sI_m - A_m)^{-1} & O_m^T \\ s^{-1}C_m(sI_m - A_m)^{-1} & s^{-1}I_q \end{bmatrix},$$

and I_m is the identity matrix with dimensions equal to that of A_m , I_q is the identity matrix with dimensions $q \times q$. By substituting the B and C matrices from (6.5), the transfer function of the augmented model is obtained as (6.18). Under the assumption that the plant model has no zero at $s = 0$ and has a minimal realization, the transfer function of the augmented model has a minimal structure from (6.5), therefore it is both controllable and observable.

6.3 Model Predictive Control Using Finite Prediction Horizon

From the analysis in Section 6.2, we understand that for a linear time invariant system, when the closed-loop system is stable, the derivative of the

¹ The zeros of a MIMO transfer function are those values of s that make the matrix $G_m(s)$ lose rank.

control signal for disturbance rejection exponentially converges to zero after the transient response period. The same is true for a set-point change. We use this observation in the design of a continuous-time model predictive control system using the receding horizon control principle.

6.3.1 Modelling the Control Trajectory

Consider a moving time window which is taken from t_i to $t_i + T_p$, and the time variable within this window is called τ to distinguish it from t , and the derivative of the control within this window is $\dot{u}(\tau)$. The augmented plant model is described by (A, B, C) matrices. With a given information of the state vector at time t_i as $x(t_i)$ for the initial state, the state feedback control, with gain matrix K is, for $0 \leq \tau \leq T_p$,

$$\dot{u}(\tau) = -Kx(\tau).$$

The closed-loop system with initial condition $x(t_i)$ is captured by

$$\begin{aligned} x(\tau) &= e^{(A-BK)\tau}x(t_i) \\ \dot{u}(\tau) &= -Ke^{(A-BK)\tau}x(t_i). \end{aligned}$$

Suppose that K is chosen such that the closed-loop system is stable with all the eigenvalues of the system matrix $(A - BK)$ strictly within the left-half complex plane. Then it is seen that the derivative $\dot{u}(\tau)$ exponentially decays to zero.

With this observation in mind, it is sufficient to assume that for every feedback control gain matrix K , there is an underlying control trajectory $\dot{u}(\tau)$, defined by a set of exponentially decaying functions within the interval $0 \leq \tau \leq T_p$. Furthermore, this underlying trajectory satisfies

$$\lim_{T_p \rightarrow \infty} \int_0^{T_p} \dot{u}(\tau)^2 d\tau < \infty. \quad (6.21)$$

From the analysis in Chapter 5, the derivative of the control signal can be described by a set of orthonormal basis functions. Namely, for $0 \leq \tau \leq T_p$,

$$\dot{u}(\tau) \approx \sum_{i=1}^N c_i l_i(\tau) = L(\tau)^T \eta, \quad (6.22)$$

where $\eta = [c_1 \ c_2 \ \dots \ c_N]^T$ is the vector of coefficients and $l_i(\tau)$, $i = 1, 2, \dots, N$ are the set of orthonormal basis functions. Both Laguerre functions and Kautz functions are the natural candidates for this task (see Chapter 5). For this description, a set of Laguerre functions is used, and for a given scaling factor $p > 0$ and some small $\epsilon > 0$, with increase of N ,

$$\int_0^\infty \left(\dot{u}(\tau) - \sum_{i=1}^N c_i l_i(\tau) \right)^2 d\tau < \epsilon,$$

which ensures the convergence of the approximation.

As discussed in Chapter 5, the Laguerre functions are best described by the state-space model (6.23), where the state vector $L(\tau) = [l_1(\tau) \ l_2(\tau) \dots \ l_N(\tau)]^T$. Assuming initial conditions of the state vector as $L(0) = \sqrt{2p} [1 \ 1 \ \dots \ 1]^T$, then the Laguerre functions satisfy the state-space equation

$$\begin{bmatrix} \dot{l}_1(\tau) \\ \dot{l}_2(\tau) \\ \vdots \\ \dot{l}_N(\tau) \end{bmatrix} = \begin{bmatrix} -p & 0 & \dots & 0 \\ -2p & -p & \dots & 0 \\ \vdots & & & \\ -2p & \dots & -2p & -p \end{bmatrix} \begin{bmatrix} l_1(\tau) \\ l_2(\tau) \\ \vdots \\ l_N(\tau) \end{bmatrix}. \quad (6.23)$$

The solution of the differential equation (6.23) gives the set of Laguerre functions for $i = 1, 2, \dots, N$ as

$$L(\tau) = e^{A_p \tau} L(0), \quad (6.24)$$

where

$$A_p = \begin{bmatrix} -p & 0 & \dots & 0 \\ -2p & -p & \dots & 0 \\ \vdots & & & \\ -2p & \dots & -2p & -p \end{bmatrix}.$$

Unlike the Laguerre functions, Kautz functions allow the flexibility of specifying the pole locations. If a set of Kautz functions is used in the description of the derivative of the control trajectory, then by choosing the set of poles of the Kautz functions to be identical to the eigenvalues of the system matrix ($A - BK$), the description can be made identical to the underlying control trajectory. Due to its increased complexity, the case that uses Kautz functions will be treated at the end of this chapter. Therefore, the focus here will be on the case that uses a set of Laguerre functions.

6.3.2 Predicted Plant Response

Assume that at the current time, say t_i , the state variable $x(t_i)$ is available. Then at the future time τ , $\tau > 0$, the predicted state variable $x(t_i + \tau | t_i)$ is described by the following equation:

$$x(t_i + \tau | t_i) = e^{A\tau} x(t_i) + \int_0^\tau e^{A(\tau-\gamma)} B \dot{u}(\gamma) d\gamma, \quad (6.25)$$

where the expected effect of the random disturbances with zero mean in the future prediction is zero. To keep notation simple, the prediction (6.25) represents both cases of deterministic and stochastic disturbance. Let the control signal be written as

$$\dot{u}(\tau) = [\dot{u}_1(\tau) \ \dot{u}_2(\tau) \ \dots \ \dot{u}_m(\tau)]^T,$$

and the input matrix be written in column form as

$$B = [B_1 \ B_2 \ \dots \ B_m],$$

where B_i is the i th column of the B matrix. The i th control signal $u_i(t)$ ($i = 1, 2, \dots, m$) is expressed as the orthonormal expansion:

$$\dot{u}_i(\tau) = L_i(\tau)^T \eta_i,$$

where the i th vector $L_i(\tau)^T$ consists of

$$L_i(\tau)^T = [l_1^i(\tau) \ l_2^i(\tau) \ \dots \ l_{N_i}^i(\tau)],$$

and the i th coefficient vector:

$$\eta_i = [c_1^i \ c_2^i \ \dots \ c_{N_i}^i]^T.$$

Here, there are a pair of p_i and N_i parameters defined for each $L_i(\tau)$.

Then, the predicted future state at time τ is

$$x(t_i + \tau | t_i) = e^{A\tau} x(t_i) + \int_0^\tau e^{A(\tau-\gamma)} [B_1 L_1(\gamma)^T \ \dots \ B_m L_m(\gamma)^T] d\gamma \eta,$$

written as

$$x(t_i + \tau | t_i) = e^{A\tau} x(t_i) + \phi(\tau)^T \eta, \quad (6.26)$$

where $\phi(\tau)^T$ is the convolution integral with

$$\phi(\tau)^T = \int_0^\tau e^{A(\tau-\gamma)} [B_1 L_1(\gamma)^T \ B_2 L_2(\gamma)^T \ \dots \ B_m L_m(\gamma)^T] d\gamma,$$

and the coefficient vector η contains all sub-coefficient vectors for each control input defined as

$$\eta^T = [\eta_1^T \ \eta_2^T \ \dots \ \eta_m^T],$$

which has dimension $\sum_{i=1}^m N_i$. Matrix $\phi(\tau)^T$ has dimension $n \times \sum_{i=1}^m N_i$.

Note that the prediction of the state variable $x(t_i + \tau | t_i)$ is captured by the set of unknown coefficients η . The integral $\phi(\tau)^T$ is uniquely determined as shown in the next section, if the orthonormal basis functions $L_1(\gamma), L_2(\gamma), \dots, L_m(\gamma)$ are chosen and the pair of matrices (A, B) are specified.

From the prediction of the state variable, the predicted output at time τ is

$$y(t_i + \tau | t_i) = C e^{A\tau} x(t_i) + C \phi(\tau)^T \eta. \quad (6.27)$$

6.3.3 Analytical Solution of the Predicted Response

The major computational load in evaluating the prediction comes from the convolution operation in (6.26), which requires the computation of $n \times \sum_{i=1}^m N_i$ integral expressions.

Let us define the convolution integral corresponding to the i th input

$$\phi_i(\tau)^T = \int_0^\tau e^{A(\tau-\gamma)} B_i L_i(\gamma)^T d\gamma, \quad (6.28)$$

where $\phi_i(\tau)^T$ is a matrix with dimensionality of $n \times N_i$, and N_i is the number of coefficients for the Laguerre functions used to describe the i th control signal. We observe that

$$x(t_i + \tau | t_i) = e^{A\tau} x(t_i) + [\phi_1(\tau)^T \phi_2(\tau)^T \dots \phi_m(\tau)^T] \eta. \quad (6.29)$$

Thus the prediction of the future state trajectory is partly expressed in terms of $\phi_i(\tau)^T$ with $1 \leq i \leq m$.

Computation of $\phi_i(\tau)^T$

Effectively, the integral expression $\phi(\tau)^T$ for an m -input system is decomposed into the computation of the individual component $\phi_i(\tau)^T$ for $1 \leq i \leq m$. This is the expression for a single input. To avoid confusion and to introduce notational simplicity in the following, we will use $\phi(\tau)^T$, p and N for the description of the i th input signal, and (A, B, C) as the system matrices. Parameters p and N may vary with different control input signals.

Proposition 6.1. *For a given τ , the matrix $\phi(\tau)^T$ satisfies the linear algebraic equation*

$$A\phi(\tau)^T - \phi(\tau)^T A_p^T = -BL(\tau)^T + e^{A\tau} BL(0)^T, \quad (6.30)$$

where $L(\tau)^T$, $L(0)^T$ and A_p are defined by (6.23) and (6.24).

Proof. From matrix differentiation, another representation of (6.28) is

$$A\phi(\tau)^T = - \int_0^\tau d(e^{A(\tau-\gamma)}) BL(\gamma)^T. \quad (6.31)$$

By applying integration by parts to the right-hand side of the above equation, we obtain

$$\begin{aligned} A\phi(\tau)^T &= -[BL(\tau)^T - e^{A\tau} BL(0)^T] \\ &\quad + \int_0^\tau e^{A(\tau-\gamma)} BL(\gamma)^T A_p^T d\gamma \end{aligned} \quad (6.32)$$

$$= -[BL(\tau)^T - e^{A\tau} BL(0)^T] + \phi(\tau)^T A_p^T, \quad (6.33)$$

where (6.24) is used to derive (6.32).

The elements in the matrix $\phi(\tau)^T$ are to be determined from the solution of $n \times N$ linear algebraic equations formed from (6.30). However, taking advantage of the lower triangular structure of the matrix A_p (see (6.24)), the elements

in $\phi(\tau)^T$ can be solved column by column in a systematic manner. Assume that we let I_i denote the i th column of $\phi(\tau)^T$ and J_i denote the i th column on the left-hand side of (6.30), we have

$$(A + pI)I_1 = J_1.$$

For $i = 2, \dots, N$,

$$(A + pI)I_i = J_i - 2p \sum_{k=1}^{i-1} I_k. \quad (6.34)$$

Through (6.34), the original solution to (6.30), which has $n \times N$ variables, is decoupled to the N separate solutions of n linear equations. This procedure enhances the numerical stability of these computations, particularly when the numbers of states and outputs are large. It is seen from (6.34) that in order to guarantee invertibility of $(A + pI)$, i.e., a unique solution for $\phi(\tau)^T$, the choice of p must satisfy $p \neq -\lambda_i(A)$ for all i , where $\lambda_i(A)$ denotes the i th eigenvalue of the matrix A .

Upon obtaining the matrices $\phi_i(\tau)^T$ for $i = 1, 2, \dots, m$, as outlined above, we finally obtain the prediction $x(t_i + \tau | t_i)$ by putting the computed matrices side by side as shown in (6.29).

6.3.4 The Recursive Solution

If the prediction $x(t_i + \tau | t_i)$ is calculated in terms of $\tau = 0, h, 2h, \dots, T_p$, then the integral expression can be solved recursively. This may prove to be useful if there are many inputs and state variables in the predictive control system. The following proposition summarizes the results.

Proposition 6.2. *For a given $h > 0$, we define*

$$\phi(h)^T = \int_0^h e^{A(h-\gamma)} BL^T(\gamma) d\gamma \quad (6.35)$$

$$\phi(2h)^T = \int_0^{2h} e^{A(2h-\gamma)} BL^T(\gamma) d\gamma \quad (6.36)$$

$\vdots \vdots \vdots$

$$\phi(kh)^T = \int_0^{kh} e^{A(kh-\gamma)} BL^T(\gamma) d\gamma. \quad (6.37)$$

Then,

$$\phi(kh)^T = e^{Ah} \phi((k-1)h)^T + \phi(h)^T e^{(k-1)A_p^T h}. \quad (6.38)$$

Proof.

$$\begin{aligned}\phi(2h)^T &= \int_0^{2h} e^{A(2h-\gamma)} BL^T(\gamma) d\gamma \\ &= e^{Ah} \left(\int_0^h e^{A(h-\gamma)} BL^T(\gamma) d\gamma + \int_h^{2h} e^{A(h-\gamma)} BL^T(\gamma) d\gamma \right) \\ &= e^{Ah} \left(\phi(h)^T + \int_h^{2h} e^{A(h-\gamma)} BL^T(\gamma) d\gamma \right).\end{aligned}\quad (6.39)$$

Note that

$$\int_h^{2h} e^{A(h-\gamma)} BL^T(\gamma) d\gamma = e^{-Ah} \int_0^h e^{A(h-\beta)} BL^T(\beta) d\beta e^{A_p^T h}, \quad (6.40)$$

where $\beta = \gamma - h$. Substituting (6.40) into (6.39), we obtain

$$\phi(2h)^T = e^{Ah} \phi(h)^T + \phi(h)^T e^{hA_p^T h}. \quad (6.41)$$

Let us assume that $\phi((k-1)h)^T$ is available, and we calculate $\phi(kh)^T$ through

$$\phi(kh)^T = \int_0^{kh} e^{A(kh-\gamma)} BL^T(\gamma) d\gamma \quad (6.42)$$

$$\begin{aligned}&= e^{Ah} \left[\int_0^{(k-1)h} e^{A((k-1)h-\gamma)} BL^T(\gamma) d\gamma \right. \\ &\quad \left. + \int_{(k-1)h}^{kh} e^{A((k-1)h-\gamma)} BL^T(\gamma) d\gamma \right].\end{aligned}\quad (6.43)$$

Note that the first term in (6.43) is $\phi((k-1)h)^T$, the second term can be expressed as

$$\int_{(k-1)h}^{kh} e^{A((k-1)h-\gamma)} BL^T(\gamma) d\gamma = e^{-Ah} \int_0^h e^{A(h-\beta)} BL^T(\beta) d\beta e^{A_p^T (k-1)h}, \quad (6.44)$$

where $\beta = \gamma - (k-1)h$ and

$$L^T(\beta + (k-1)h) = L^T(\beta) e^{A_p^T (k-1)h}$$

because of the exponential nature of the Laguerre functions.

By substituting (6.44) into (6.43), (6.38) is shown to be true.

Example 6.2. Assume a first-order system with scalars $A = a$ and $B = b$. A single Laguerre function is used with $N = 1$ and scaling factor p . Evaluate the convolution integral expression using the recursive approach and the direct computation via Proposition 6.2.

Solution. For a given sampling interval h , through straightforward calculation, we have

$$\begin{aligned}\phi(h)^T &= \int_0^h e^{a(h-\gamma)} b l(0) e^{-p\gamma} d\gamma = \frac{b\sqrt{2p}}{a+p} [e^{ah} - e^{-ph}] \\ \phi(2h)^T &= \int_0^{2h} e^{a(2h-\tau)} b l(0) e^{-p\tau} d\tau = \frac{b\sqrt{2p}}{a+p} [e^{2ah} - e^{-2ph}],\end{aligned}$$

and so on, then leading to

$$\phi(kh)^T = \int_0^{2h} e^{a(kh-\tau)} b l(0) e^{-p\tau} d\tau = \frac{b\sqrt{2p}}{a+p} [e^{akh} - e^{-pkh}]. \quad (6.45)$$

Using the results presented in Proposition 6.1, we can also come to the same expression of $\phi(kh)^T$ through

$$\begin{aligned}\phi(kh)^T &= e^{ah} \phi((k-1)h)^T + \phi(h)^T e^{-(k-1)hp} \\ &= \frac{b\sqrt{2p}}{a+p} [e^{akh} - e^{(-(k-1)p+a)h} + e^{(-(k-1)p+a)h} - e^{-pkh}] \\ &= \frac{b\sqrt{2p}}{a+p} [e^{akh} - e^{-pkh}].\end{aligned} \quad (6.46)$$

The results in Proposition 6.2 simplify the computational procedure to obtain the gain matrices for the continuous-time model predictive control. Instead of solving a set of linear equations at every discrete time, the integral expression is solved through a simple recursive procedure. Also, note that h is the discretization interval within the optimization window. This is not related to the actual sampling interval in the implementation of the predictive control system. In general, h can be very small to approximate the continuous-time prediction as accurately as possible.

The first quantity in the recursive calculation requires the solution of the linear equations. The following tutorial demonstrates how to solve the algebraic matrix equation (6.30) effectively.

Tutorial 6.1. *The objective of this tutorial is to demonstrate how to obtain the integral expression $\phi(\tau)^T$ by solving the linear algebraic equation*

$$AX - XA_p^T = Y, \quad (6.47)$$

where A_p is defined by the system matrix used to generate the set of Laguerre functions (see (6.24)).

Step by Step

1. Create a new file called *Int.m*. This function will be used to solve the linear algebraic equation (6.47) with matrices A , A_p and Y . We will find the solution recursively by taking advantage of the lower triangular structure of the matrix A_p .
2. Enter the following program into the file:

```

function X=Iint(A,p,Y)
[n,N]=size(Y);
X=zeros(n,N);
Ai=inv(A+p*eye(n,n));
X(:,1)=Ai*Y(:,1);
alpha=-2*p*X(:,1);
for i=2:N;
    X(:,i)=Ai*(Y(:,i)+alpha);
    alpha=alpha-2*p*X(:,i);
end

```

3. Write a test program to verify if the results are correct, based on the first-order system of Example 6.2.
4. This function is useful later as part of the design program for the continuous-time model predictive control.

6.4 Optimal Control Strategy

The Cost function

Similar to the discrete-time case, in the continuous-time predictive control, the cost function is also assumed to take the form:

$$J = \int_0^{T_p} (x(t_i + \tau | t_i)^T Q x(t_i + \tau | t_i) + \dot{u}(\tau)^T R \dot{u}(\tau)) d\tau, \quad (6.48)$$

with initial state information given as $x(t_i)$. The optimal control $\dot{u}(\tau)$ is found by minimizing this cost function J , which is the optimal control within the moving window. The question arises as to how the set-point signal should enter the optimization. The optimal performance is specified by selecting weight matrices $Q \geq 0$ and $R \geq 0$.

In the traditional predictive control design, at time t_i , the cost function is often chosen as

$$J = \int_0^{T_p} ((r(t_i) - y(t_i + \tau | t_i))^T (r(t_i) - y(t_i + \tau | t_i)) + \dot{u}(\tau)^T R \dot{u}(\tau)) d\tau. \quad (6.49)$$

Without constraints, the objective of model predictive control in the case of set-point following is to find the control law that will drive the predicted plant output $y(t_i + \tau | t_i)$ as close as possible, in a least squares sense, to the future trajectory of the set-point $r(t_i)$. The assumption is that the set-point signal $r(t_i)$ is a constant (or a set of constants) within the optimization window.

When the set-point is a constant signal, by subtracting $r(t_i)$ from the variable $y(t_i + \tau | t_i)$, then the augmented model takes the form:

$$\begin{bmatrix} \dot{z}(t_i + \tau | t_i) \\ \dot{e}(t_i + \tau | t_i) \end{bmatrix} = \begin{bmatrix} A_m & o_m^T \\ C_m & o_{q \times q} \end{bmatrix} \begin{bmatrix} z(t_i + \tau | t_i) \\ e(t_i + \tau | t_i) \end{bmatrix} + \begin{bmatrix} B_m \\ o_{q \times m} \end{bmatrix} \dot{u}(\tau), \quad (6.50)$$

where $e(t_i + \tau \mid t_i) = y(t_i + \tau \mid t_i) - r(t_i)$, and $r(t_i)$ is a constant vector for $0 \leq \tau \leq T_p$, defined by

$$r(t_i) = [r_1(t_i) \ r_2(t_i) \ \dots \ r_q(t_i)]^T.$$

Note that $C = [o_m \ I_{q \times q}]$. By choosing $Q = C^T C$, in conjunction with the augmented model (6.50), the traditional cost function (6.49) can be written in the form

$$J = \int_0^{T_p} (x(t_i + \tau \mid t_i)^T Q x(t_i + \tau \mid t_i) + \dot{u}(\tau)^T R \dot{u}(\tau)) d\tau, \quad (6.51)$$

where the initial state variable information $x(t_i)$ contains the error $y(t_i) - r(t_i)$, instead of $y(t_i)$.

Solution of optimal control

We may work directly with the cost function (6.51). However, for simplicity, we assume that R is a diagonal matrix with

$$R = \text{diag}\{r_k\}, \quad (6.52)$$

where $k = 1, 2, \dots, m$.

Then, the second term in the cost function (6.51) is

$$\int_0^{T_p} \dot{u}(\tau)^T R \dot{u}(\tau) d\tau = \sum_{k=1}^m \int_0^{T_p} r_k \dot{u}_k(\tau)^2 d\tau. \quad (6.53)$$

Note, in common practice, the prediction horizon T_p is chosen to be sufficiently large such that $\dot{u}(\tau) \approx 0$ for $\tau \geq T_p$. Namely, the prediction horizon is selected to be larger than the time for which the control signal is effective. Thus,

$$\int_0^{T_p} \dot{u}_k(\tau)^T \dot{u}_k(\tau) d\tau \approx \int_0^\infty \eta_k^T L_k(\tau) L_k(\tau)^T \eta_k d\tau = \eta_k^T \eta_k, \quad (6.54)$$

where we have taken advantage of the orthonormal property of the Laguerre functions (see Chapter 5) that $\int_0^\infty L_k(\tau) L_k(\tau)^T d\tau$ is the identity matrix with dimension equal to the number of Laguerre coefficients for the k th input. The cost function J is then equivalently given by

$$J = \int_0^{T_p} x(t_i + \tau \mid t_i)^T Q x(t_i + \tau \mid t_i) d\tau + \eta^T R_L \eta, \quad (6.55)$$

where R_L is a block diagonal matrix with the k th block being R_k , and $R_k = r_k I_{N_k \times N_k}$ (where $I_{N_k \times N_k}$ is a unit matrix with dimensions $N_k \times N_k$).

Minimization of the cost function

By substituting the prediction of the state variable $x(t_i + \tau | t_i)$ from (6.26) into (6.55), the cost function becomes

$$J = \int_0^{T_p} (e^{A\tau} x(t_i) + \phi(\tau)^T \eta)^T Q (e^{A\tau} x(t_i) + \phi(\tau)^T \eta) d\tau + \eta^T R_L \eta, \quad (6.56)$$

which is a quadratic function of η

$$\begin{aligned} J &= \eta^T \left[\int_0^{T_p} \phi(\tau) Q \phi(\tau)^T d\tau + R_L \right] \eta + 2\eta^T \int_0^{T_p} \phi(\tau) Q e^{A\tau} d\tau x(t_i) \\ &\quad + x(t_i)^T \int_0^{T_p} e^{A^T \tau} Q e^{A\tau} d\tau x(t_i). \end{aligned} \quad (6.57)$$

For notational simplicity, we define

$$\Omega = \int_0^{T_p} \phi(\tau) Q \phi(\tau)^T d\tau + R_L \quad (6.58)$$

$$\Psi = \int_0^{T_p} \phi(\tau) Q e^{A\tau} d\tau. \quad (6.59)$$

Completing the square of (6.57) leads to

$$\begin{aligned} J &= [\eta + \Omega^{-1} \Psi x(t_i)]^T \Omega [\eta + \Omega^{-1} \Psi x(t_i)] \\ &\quad + x(t_i)^T \int_0^{T_p} e^{A^T \tau} Q e^{A\tau} d\tau x(t_i) - x(t_i)^T \Psi^T \Omega^{-1} \Psi x(t_i). \end{aligned}$$

Hence, the optimal η that minimizes J is, as the last two terms are independent of η

$$\eta = -\Omega^{-1} \Psi x(t_i), \quad (6.60)$$

and the minimum of the cost function is

$$J_{min} = x(t_i)^T \left[\int_0^{T_p} e^{A^T \tau} Q e^{A\tau} d\tau - \Psi^T \Omega^{-1} \Psi \right] x(t_i). \quad (6.61)$$

Once η is found, the whole trajectory of $\dot{u}(\tau)$ can be constructed using the Laguerre functions as

$$\dot{u}(\tau) = \begin{bmatrix} L_1(\tau)^T & o_2 & \dots & o_m \\ o_1 & L_2(\tau)^T & \dots & o_m \\ \vdots & \vdots & \ddots & \vdots \\ o_1 & o_2 & \dots & L_m(\tau)^T \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_m \end{bmatrix}, \quad (6.62)$$

where the row vector o_k contains zero elements and has a dimension identical to $L_k(\tau)^T$.

Computation of Ω and Ψ Matrices

The matrices Ω and Ψ are constant matrices defined by the integral expressions (6.58) and (6.59). They are computed off-line. In general, it is difficult to obtain the analytical solutions for the integral expressions that produce Ω and Ψ . However, noting that these matrices are computed over a given prediction horizon T_p , the integral expressions can be evaluated off-line using a numerical approximation scheme. More specifically, letting $\tau = 0, h, 2h, \dots, Mh$ with a constant step size h , we have the approximate relations as below:

$$\Omega \approx \sum_{k=0}^M \phi(kh)Q\phi(kh)^T h + R_L \quad (6.63)$$

$$\Psi \approx \sum_{k=0}^M \phi(kh)Qe^{Ak^h}h, \quad (6.64)$$

where $\phi(kh)^T$ is computed recursively using (6.38). The step size h in the numerical scheme would typically be chosen related to the scaling factor p , ideally as small as possible, but independent of the actual sampling interval of the process.

6.5 Receding Horizon Control

With the optimal coefficient vector η , the derivative of the future control signal is obtained using (6.62) for all $0 \leq \tau \leq T_p$.

By applying the principle of receding horizon control (*i.e.*, the control action will use only the derivative of the future control signal at $\tau = 0$), the derivative of the optimal control for the unconstrained problem with finite horizon prediction is

$$\dot{u}(t_i) = \begin{bmatrix} L_1(0)^T & o_2 & \dots & o_m \\ o_1 & L_2(0)^T & \dots & o_m \\ \vdots & \vdots & \ddots & \vdots \\ o_1 & o_2 & \dots & L_m(0)^T \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_m \end{bmatrix}. \quad (6.65)$$

For an arbitrary time t , $\eta = -\Omega^{-1}\Psi x(t)$, the continuous-time derivative of the control is

$$\begin{aligned} \dot{u}(t) &= - \begin{bmatrix} L_1(0)^T & o_2 & \dots & o_m \\ o_1 & L_2(0)^T & \dots & o_m \\ \vdots & \vdots & \ddots & \vdots \\ o_1 & o_2 & \dots & L_m(0)^T \end{bmatrix} \Omega^{-1}\Psi x(t) \\ &= -K_{mpc}x(t), \end{aligned} \quad (6.66)$$

where the feedback gain matrix is

$$K_{mpc} = \begin{bmatrix} L_1(0)^T & o_2 & \dots & o_m \\ o_1 & L_2(0)^T & \dots & o_m \\ \vdots & \vdots & \ddots & \vdots \\ o_1 & o_2 & \dots & L_m(0)^T \end{bmatrix} \Omega^{-1} \Psi.$$

It is seen from (6.66) that the receding horizon control law is in the nature of state feedback control because of the dependence on the current state variable $x(t)$. Note that the augmented state-space model is

$$\dot{x}(t) = Ax(t) + Bu(t).$$

Using this augmented state-space model where the input is $\dot{u}(t)$, we obtain the closed-loop control system as

$$\dot{x}(t) = (A - BK_{mpc})x(t), \quad (6.67)$$

from which the closed-loop eigenvalues of the predictive control system can be evaluated.

Here, we only need to integrate to reveal the control law and the action of integral control,

$$u(t) = \int_0^t \dot{u}(\tau) d\tau. \quad (6.68)$$

Note that the state variable $x(t)$ consists of two components. The first component is the derivative of the plant state $x_m(t)$. With a set-point signal $r(t)$, the error signal $y(t) - r(t)$ is the second part of the feedback component. Thus, we can write the state feedback control as

$$\dot{u}(t) = -K_{mpc}x(t) = -[K_x \ K_y] \begin{bmatrix} \dot{x}_m(t) \\ y(t) - r(t) \end{bmatrix}.$$

Figure 6.2 illustrates the block diagram of the predictive control system, where we can clearly see that it has embedded integral action.

MATLAB Tutorial: Continuous-time Predictive Control

Tutorial 6.2. *The objective of this tutorial is to demonstrate how to calculate Ω and Ψ matrices in the cost function*

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(t_i),$$

used for the design of continuous-time model predictive control. The state-space model is captured by the pair (A, B) matrices; the weight matrices are Q and R ; the prediction horizon is T_p , and the number of Laguerre parameters

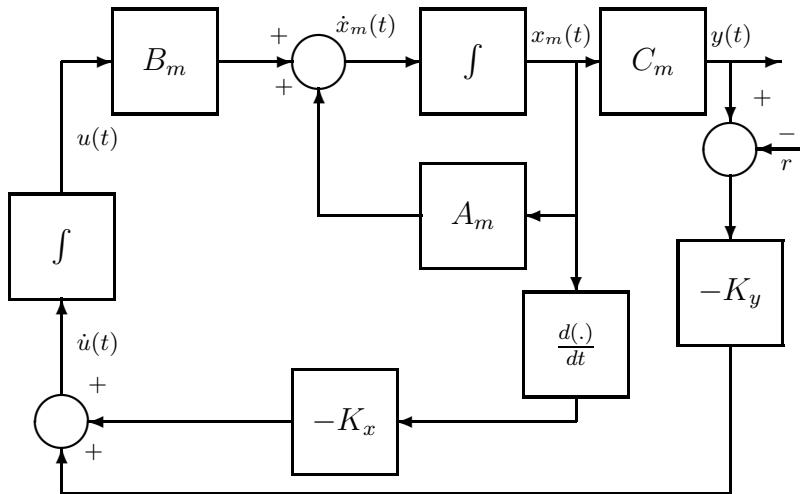


Fig. 6.2. Block diagram of continuous-time predictive control system

for each input is specified as an element in N , with a corresponding scaling parameter in p . Both N and p have dimension equal to the number of inputs.

Step by Step

1. Create a new file called *cmpc.m*.
2. This function will be used as the basis for the design of a continuous-time model predictive control system. The input parameters to the function are the state-space model, scaling factor and number of terms in the Laguerre functions, prediction horizon in a time unit, and the weight matrices Q and R . In the first part of the program, the initialization of the parameter matrices are sought. The continuous-time window is discretized in order to compute the integral expressions numerically.
3. Enter the following program into the file:

```
function [Omega,Psi]=cmpc(A,B,p,N,Tp,Q,R);
[n,n_in]= size(B);
tau_del=0.001/max(p);
Tpm=max(Tp);
tau=0:tau_del:Tpm;
Np=length(tau);
N_pa=sum(N);
Omega=zeros(N_pa,N_pa);
Psi=zeros(N_pa,n);
S_in=zeros(n,N_pa);
```

4. R_L is the weight matrix corresponding to the weight on the coefficient vector η , which is directly related to the original weight R on the input signal \dot{u} .

5. Continue entering the following program into the file:

```
R_L=eye(N_pa,N_pa);
kk=1;
for i=1:n_in
R_L(kk:kk-1+N(i),kk:kk-1+N(i))=
R(i,i)*R_L(kk:kk-1+N(i),kk:kk-1+N(i));
kk=kk+N(i);
end
```

6. The computation of $\phi(\tau)^T$ at the first sample $\tau = \tau_{\text{del}}$ is performed by solving the linear algebraic matrix equation using the function *Iint.m* that we created before. The integral expression corresponding to the first input is solved first, and the rest of the integral expressions are computed input by input and put together to form S_{sum} . *lage.m* is the MATLAB function for generating the continuous-time Laguerre system matrix and initial condition, which can be found in Tutorial 5.1.

7. Continue entering the following program into the file:

```
[A1,L0]=lagc(p(1),N(1));
Eae=expm(A*tau_del);
Eap=expm(A1*tau_del);
L=Eap*L0;
Y=-B(:,1)*L'+Eae*B(:,1)*L0';
X=Iint(A,p(1),Y);
S_in(:,1:N(1))=X;
In_s=1;
for jj=2:n_in;
[A1,L0]=lagc(p(jj),N(jj));
Eap=expm(A1*tau_del);
L=Eap*L0;
Y=-B(:,jj)*L'+Eae*B(:,jj)*L0';
X=Iint(A,p(jj),Y);
In_s=N(jj-1)+In_s;
In_e=In_s+N(jj)-1;
S_in(:,In_s:In_e)=X;
end
S_sum=S_in;
```

8. S_{sum} is the initial condition for the recursive computation of the convolution integral expression. From this point onwards, the integral expression is computed recursively.

9. Continue entering the following program into the file:

```

for i=2:Np-1;
kk=1;
[A1,L0]=lagc(p(kk),N(kk));
Eap=expm(A1*tau_del);
S_sum(:,1:N(kk))=Eae*S_sum(:,1:N(kk))
+S_in(:,1:N(kk))*(Eap^(i-1))';
In_s=1;
for kk=2:n_in;
[A1,L0]=lagc(p(kk),N(kk));
Eap=expm(A1*tau_del);
In_s=N(kk-1)+In_s;
In_e=In_s+N(kk)-1;
S_sum(:,In_s:In_e)=Eae*S_sum(:,In_s:In_e)+%
S_in(:,In_s:In_e)*(Eap^(i-1))';
end
phi=S_sum;
Omega=Omega+phi'*Q*phi;
Psi=Psi+phi'*Q*Eae^i;
end

```

10. Finally multiplying the sampling interval τ_{del} and adding the weight matrix R_L gives the Ω and Ψ matrices. Continue entering the following program into the file:

```

Omega=Omega*tau_del+R_L;
Psi=Psi*tau_del;

```

The function cmpc.m needs to be tested before further application. This test will be based on comparing results of the feedback gain matrices and closed-loop eigenvalues when using CMPC and using LQR design.

Example 6.3. Suppose that a two-input and two-output system with three states is described by the state-space model

$$\dot{x}_m(t) = A_m x_m(t) + B_m u(t); \quad y(t) = C_m x(t),$$

where the system matrices are

$$A_m = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -3 & 0 \\ 3 & 3 & -5 \end{bmatrix}; \quad B_m = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}; \quad C_m = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Augment this model with integrators and design a continuous-time predictive control with the specified parameters as follows, $N_1 = N_2 = 4$; and $p_1 = 1.5$; $p_2 = 2$; $T_p = 10$; $Q = C^T C$ and $R = 0.2I$. Calculate the continuous-time predictive control gain matrix K_{mpc} and the closed-loop eigenvalues of the predictive control system. Compare the continuous-time MPC results with

the results obtained using MATLAB LQR function.

Solution.

Step by Step

1. Create a program called *testcmmpc.m*
2. We will first form the augmented state-space model and set the design parameters in the continuous-time predictive control.
3. Enter the following program into the file:

```

Am=[-1 0 0;0 -3 0;3 3 -5];
Bm=[1 0;0 1; 1 1];
Cm=[1 0 0;0 1 0];
[m1,n1]=size(Cm);
[n1,n_in]=size(Bm);
A=zeros(n1+m1,n1+m1);
A(1:n1,1:n1)=Am;
A(n1+1:n1+m1,1:n1)=Cm;
B=zeros(n1+m1,n_in);
B(1:n1,:)=Bm;
C=zeros(m1,n1+m1);
C(:,n1+1:n1+m1)=eye(m1,m1);
Q=C'*C;
R=0.2*eye(2,2);
p1=1.5;
p2=2;
N1=4;
N2=4;
p=[p1 p2];
N=[N1 N2];
Tp=10;

```

4. Call the function *cmmpc.m* created from Tutorial 6.2 by entering the following line into the file:

```
[Omega,Psi]=cmmpc(A,B,p,N,Tp,Q,R);
```

5. We need to recover the state feedback control gain matrix and the closed-loop eigenvalues. In order to do so, the initial condition of the Laguerre functions needs to be used. Matrix *Lzerot* is used for realization of the feedback gain matrix.

6. Continue entering the following program into the file:

```

[Ap1,L1]=lagc(p1,N1);
[Ap2,L2]=lagc(p2,N2);
Lzerot=zeros(2,N1+N2);
Lzerot(1,1:N1)=L1';

```

```
Lzerot(2,N1+1:N1+N2)=L2';
```

7. We are ready to calculate the state feedback gain matrix and the closed-loop eigenvalues. Continue entering the following program into the file:

```
K_mpc=Lzerot*(Omega\Psi);
Acl=A-B*K_mpc;
eig(Acl)
```

8. Finally we should compare the results with the LQR results. Enter the following line into the file:

```
[K,S,E]=lqr(A,B,Q,R);
```

9. Run this test program.

The state feedback control gain matrices of CMPC and LQR are shown in Table 6.1. It is seen that the elements in the state feedback control gain matrix from CMPC are crude approximations to those from LQR. The same can be said for the elements of the closed-loop eigenvalues. It will be shown in Chapter 8 that the numerical instability of the predictive control algorithm is the main cause of the discrepancy and the accuracy of the approximation will improve if exponential data weighting is used in the design.

Table 6.1. Comparison of numerical results between LQR and CMPC

	State feedback gain				Closed-loop eigenvalues			
CMPC	1.3017	0	0	2.1838	0	-5	-2.8993	-0.7233 - 1.1508 ± j0.9270
LQR	1.3393	0	0	2.2361	0	-5	-2.8992	-0.7713 - 1.1696 ± j0.9317

Example 6.4. Consider the state-space model

$$\begin{aligned}\dot{x}_m(t) &= \begin{bmatrix} 0 & 1 \\ -1 & -3 \end{bmatrix} x_m(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x_m(t).\end{aligned}\quad (6.69)$$

The augmented model for the design of model predictive control takes the form:

$$\begin{aligned}\dot{x}(t) &= \begin{bmatrix} 0 & 1 & 0 \\ -1 & -3 & 0 \\ 1 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \dot{u}(t) \\ y(t) &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x(t).\end{aligned}\quad (6.70)$$

Within one optimization window, examine the trajectory of the model predictive control as an approximation of the true optimal solution from linear quadratic regulator (LQR).

Solution. Since the design principle is to model the optimal derivative of control trajectory using Laguerre functions, we examine the accuracy of the approximation for a chosen set of weight matrices Q and R . For simplicity, we choose $Q = I$ (*i.e.*, a 3×3 identity matrix) and $R = 0.1$. Using MATLAB function *lqr*, the optimal controller K is obtained via minimizing the cost function:

$$J = \int_0^{\infty} (x(t)^T Q x(t) + \dot{u}(t)^T R \dot{u}(t)) dt. \quad (6.71)$$

Table 6.2 shows the comparative results between the optimal control system designed by solving the algebraic Riccati equation and the predictive control system using Laguerre functions.

Figure 6.3 shows the comparative results of the closed-loop response between LQR and the predictive control systems (*i.e.*, $N = 1, 2, 3$) with an identical initial state variable condition. Two sets of initial state variable conditions are examined. It is seen that the approximation of the derivative of the control signal is indeed very close to the optimal signal, even with $N = 1$. As N increases, the accuracy improves. As a result, the output response is very close to the response from LQR system for higher values of N .

Table 6.2. Comparison between LQR and predictive control

	State feedback gain	Closed-loop eigenvalues
LQR	5.7796 2.5280 3.1623	-4.0454 - 0.7413 $\pm j0.4818$
MPC ($N = 1$)	5.7494 1.9016 2.5278	-2.8360 - 1.4515 - 0.6141
MPC ($N = 2$)	5.2423 2.1431 3.2292	-3.6879 - 0.7276 $\pm j0.5884$
MPC ($N = 3$)	5.3663 2.3530 2.8289	-3.9097 - 0.7216 $\pm j0.4503$

This example shows that within one optimization window, the derivative of the control trajectory converges to the underlying optimal control dictated by the linear feedback control law, which is the linear quadratic regulator solution with the same Q and R matrices.

6.6 Implementation of the Control Law in Digital Environment

6.6.1 Estimation of the States

The prediction of the future plant behaviour is dependent on the availability of the state variable $x(t_i)$ at time t_i . In general, $x(t_i)$ is not measurable. In

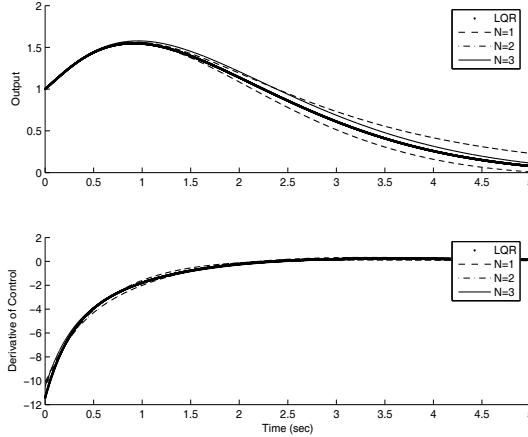


Fig. 6.3. Comparison study between LQR and predictive control, $x_1(0)=x_2(0)=x_3(0)=1$

this particular formulation, $x(t)$ contains the derivative of the original plant state variable $x_m(t)$. Because the derivative operation will amplify the high frequency noise in the system, it is not recommended to obtain the derivative functions without a filter. A continuous-time observer will play the dual role, which involves accessing the plant state variable information and filtering the measurement noise.

The observer equation that is needed for implementing the continuous-time MPC that has embedded integrators is given by

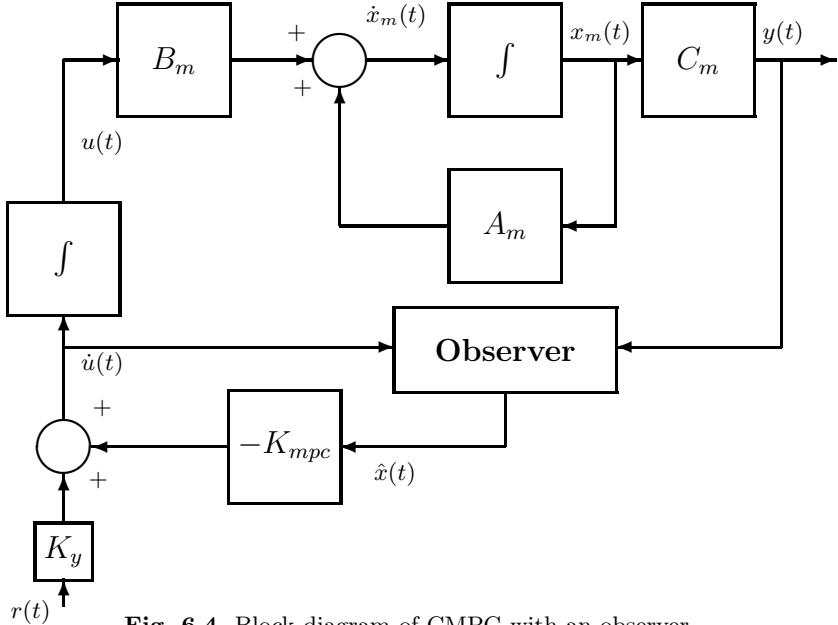
$$\frac{d\hat{x}(t)}{dt} = A\hat{x}(t) + B\dot{u}(t) + K_{ob}(y(t) - C\hat{x}(t)), \quad (6.72)$$

where $\hat{x}(t)$ is the estimate of $x(t)$, K_{ob} is the observer gain, and (A, B, C) are the system matrices for the augmented model. Signal $\dot{u}(t)$ is determined from the optimal solution of the model predictive control strategy with integrator embedded. Provided that the system model (A, B, C) is completely observable, in theory, K_{ob} can be chosen such that the error, $\tilde{x}(t) = x(t) - \hat{x}(t)$, decays exponentially at a desired rate. However, in practice, the observer gain K_{ob} is often limited by the presence of measurement noise and is selected based on the disturbance characteristics of the process. The continuous-time Kalman filters are often used in the noise environment to estimate the state variable $\hat{x}(t)$ (Anderson and Moore, 1979; Grimble and Johnson, 1988b).

Figure 6.4 shows the block diagram of the closed-loop feedback system using an observer. Note that the set-point signal is weighted by the feedback gain K_y , which was the gain matrix applied to the output $y(t)$ in Figure 6.2.

The observer (6.72) is implemented in a discretized-form as

$$\hat{x}(t_i + \Delta t) = \hat{x}(t_i) + (A\hat{x}(t_i) + B\dot{u}(t_i) + K_{ob}(y(t_i) - C\hat{x}(t_i))) \Delta t. \quad (6.73)$$

**Fig. 6.4.** Block diagram of CMPC with an observer

Thus, in conjunction with the information of $\dot{u}(t_i)$, obtained from predictive control, and $y(t_i)$, $x(t_i)$, the next sample of state estimate $\hat{x}(t_i + \Delta t)$ is computed.

The advantage of the proposed control system structure lies in the ease of implementation. First, the steady state of the state vector $\dot{x}_m(t)$ is zero, so is the steady state of $\dot{u}(t)$. Thus, these steady-state values do not enter the implementation. In addition, the control calculation is carried out by simply approximating the derivative $\dot{u}(t)$ with

$$\dot{u}(t_i) \approx \frac{u(t_i) - u(t_i - \Delta t)}{\Delta t}. \quad (6.74)$$

Hence,

$$u(t_i) = u(t_i - \Delta t) + \dot{u}(t_i)\Delta t. \quad (6.75)$$

Assume that the steady state value of the control signal is u_{ss} , then the actual control signal to the plant is

$$u_{act}(t_i) = u_{ss} + u(t_i) = u_{ss} + u(t_i - \Delta t) + \dot{u}(t_i)\Delta t. \quad (6.76)$$

By taking $u_{act}(t_i - \Delta t) = u_{ss} + u(t_i - \Delta t)$, then the actual control signal to the plant is calculated as

$$u_{act}(t_i) = u_{act}(t_i - \Delta t) + \dot{u}(t_i)\Delta t. \quad (6.77)$$

In this way, the involvement of a steady-state value of the control signal is avoided. Instead, the actual control signal will repeatedly update itself from its past value.

In the receding horizon control, we construct the derivative $\dot{u}(t_i)$ through

$$\dot{u}(t_i) = \begin{bmatrix} L_1(0)^T & o_2 & \dots & o_m \\ o_1 & L_2(0)^T & \dots & o_m \\ \vdots & \vdots & \ddots & \vdots \\ o_1 & o_2 & \dots & L_m(0)^T \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_m \end{bmatrix}. \quad (6.78)$$

Finally,

$$u_{act}(t_i) = u_{act}(t_i - \Delta t) + \begin{bmatrix} L_1(0)^T & o_2 & \dots & o_m \\ o_1 & L_2(0)^T & \dots & o_m \\ \vdots & \vdots & \ddots & \vdots \\ o_1 & o_2 & \dots & L_m(0)^T \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_m \end{bmatrix} \Delta t. \quad (6.79)$$

Equation (6.79) is the so called control signal in velocity form. If we set $u_{act}(0) = u_{ss}$ at the initial time, then the actual control signal will be automatically updated as each new value of the derivative of the control signal is computed.

6.6.2 MATLAB Tutorial: Closed-loop Simulation

Tutorial 6.3. *The objective of this tutorial is to demonstrate how to perform a simulation of a closed-loop continuous-time model predictive control system without constraints. An observer is used in the simulation.*

Step by Step

1. Create a new file called *cssimuob.m*.
2. This function will be used as the basis for the simulation of a continuous-time model predictive control system without constraints. The input parameters to the function are the initial conditions of the plant state $xm(0)$, control signal $u(-h)$ (h is the sampling interval for the continuous-time system), output signal $y(0)$. Set-point signal for the entire simulation, sp , where sp is defined as a data matrix with the number of rows corresponding to the number of outputs and the number of columns corresponding to the number of data points for the set-point signal. Ap , Bp and Cp correspond to the plant model; A , B and C correspond to the augmented model used in the design of a continuous-time predictive control system. The plant model may be different from the model used for design, permitting an assessment of robustness of the predictive control system. N_{sim} is the number of simulation time steps in discrete points. Ω and Ψ are the gain matrices in the predictive control system. K_{ob} is the observer gain,

Lzerot is the data matrix for the initial condition of the Laguerre functions in transposed form. Because the problem is formulated in continuous-time, the sampling interval *h* can also change during the simulation to simulate the case of irregular sampling.

3. Enter the following program into the file:

```
function [u1,y1,udot1,t]=cssimuob(xm,u,y,sp,Ap,Bp,Cp,
A,B,C,N_sim,Omega,Psi,K_ob,Lzerot,h)
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
X_hat=zeros(n1+m1,1);
```

4. The above program finds the number of inputs and number of outputs, and defines the initial condition of the estimated state variable.
5. The simulation is performed recursively. At time *kk*, the set-point signal enters the simulation through the difference between *X_hat(kk)* and *Xsp*, where *Xsp* is a vector having the number of zeros corresponding to the number of states, the rest of it corresponds to the set-point signal at time *kk*. The Laguerre coefficient vector *eta* (η) is found using the closed-form expression: $\eta = -\Omega^{-1}\Psi(\hat{x}(t_k) - x_{sp}(t_k))$. The derivative of the control signal is constructed using the value of *eta* and the initial Laguerre data matrix *Lzerot*. The control signal is updated with its past value and the current derivative value.

6. Continue entering the following program into the file:

```
for kk=1:N_sim;
Xsp=[zeros(n1,1);sp(:,kk)];
eta=-(Omega\Psi)*(X_hat-Xsp);
udot=Lzerot*eta;
u=u+udot*h;
```

7. Store the signals for plotting later, and update the observed state variable with *y* and new *udot*.
8. Continue entering the following program into the file:

```
udot1(1:n_in,kk)=udot;
u1(1:n_in,kk)=u;
y1(1:m1,kk)=y;
X_hat=X_hat+(A*X_hat+K_ob*(y-C*X_hat))*h+B*udot*h;
```

9. Using the current control *u*, the plant state and output are updated. Continue to enter the following program into the file:

```
xm=xm+(Ap*xm+Bp*u)*h;
y=Cp*xm;
end
t=0:h:(N_sim-1)*h;
```

10. In the real-time implementation of continuous-time MPC, the computation of the plant output is replaced by the measurement of the plant output and the input is sent to the plant input.
11. If there is measurement noise or input disturbance, they can be easily added to the simulation. For instance, if the measurement noise sequence is epsilon , then we will change the output to

```
y=Cp*xm+E*epsilon(kk);
```

where E is used to indicate which output will have the noise. For example, if the measurement noise is added to the first output, then E is equal to one for the first element and zero for the rest. Similarly, if there are input disturbances to the plant, then we will add the input disturbances to the control signal in the simulation.

This simulation program will be used to assess performance for the predictive control system without constraints. It needs to be tested. A test example is given as follows.

Example 6.5. Consider a two-input, two-output system described by the transfer function

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{12.8(-s+4)^2}{(16.7s+1)(s+4)^2} & \frac{-10.9(-3s+4)^2}{(21.0s+1)(3s+4)^2} \\ \frac{12.8(-7s+4)^2}{(10.9s+1)(7s+4)^2} & \frac{-19.4(-3s+4)^2}{(14.4s+1)(3s+4)^2} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}. \quad (6.80)$$

In the simulation, a unit-step reference signal is applied to y_1 and the reference signal to y_2 is zero. The Laguerre parameters are selected as $N1 = N2 = 2$; and $p1 = 0.5/15$; $p2 = 0.5/26$. $Q = C^T C$ and $R = 0.2I$. The prediction horizon is selected as $T_p = 50$. The observer is designed using MATLAB program LQR where the weight matrices $Q_1 = I$ and $R_1 = 0.0001 \times I$. The sampling interval in the simulation is selected as 0.001 sec.

Solution. Create a program called testsim.m. Following the steps in the example with the appropriate system parameters and performance parameter specifications leads to the computation of the Ω and Ψ matrices. To calculate the observer gain matrix, enter the following lines into the file:

```
Q1=eye(n1+m1,n1+m1);
R1=0.0001*eye(m1,m1);
K_ob=lqr(A',C',Q1,R1)';
```

Also, enter the initial conditions for the simulation by entering the following lines into the file:

```
xm=zeros(n1,1);
u0=zeros(n_in,1);
y0=zeros(m1,1);
N_sim=60000;
```

```

h=0.001;
r1=ones(1,N_sim+200);
r2=zeros(1,N_sim+200);
sp=[r1;r2];

```

The function from Tutorial 6.3 is called to produce closed-loop simulation results. Enter the following line into the file:

```

[u,y,udot,t]=cssimuob(xm,u0,y0,sp,Ap,Bp,Cp,A,B,C,N_sim,
Omega,Psi,K_ob,Lzerot,h);

```

Figure 6.5 shows the output signal y_1 and y_2 and the closed-loop control signal u_1 and u_2 . It is seen from this figure that the closed-loop system is stable, and due to the integral action, the output y_1 has followed the set-point signal without steady-state error, while the output y_2 is equal to zero at steady state.

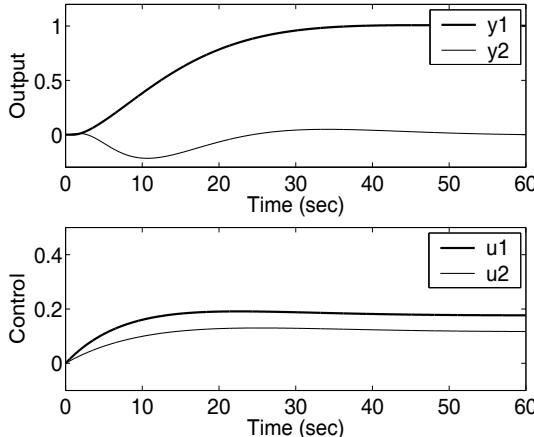


Fig. 6.5. Closed-loop system response

6.7 Model Predictive Control Using Kautz Functions

Until now, the orthonormal basis functions used in the design were Laguerre functions. For a set of Laguerre functions, there are two pre-chosen parameters. One is its pole location p , while the other is the number of terms, N , to be used in the orthonormal expansion. We understand that for a pair of Q and R matrices, there is an underlying control trajectory that is the optimal control of the LQR system with given initial condition $x(t_i)$. Therefore, the best choice of p should correspond to the dominant pole of the closed-loop system $A - BK_{lqr}$. With this choice, increasing the number of terms N will

lead to convergence of the predictive control trajectory to the optimal control trajectory of LQR within one optimization window. The question arises as to whether there is another set of orthonormal functions that will precisely capture the optimal LQR control trajectory with a finite number of terms corresponding to the number of state variables. The answer to this question leads to the application of Kautz functions in the design of predictive control. Although the solution is presented in continuous time, the same procedure can be extended to discrete time when using discrete-time Kautz functions.

Kautz functions are also orthonormal functions, which have been discussed in Chapter 5. They are a generalization of Laguerre functions, allowing both real and complex poles to be used in its Laplace transfer function networks.

Similar to the application of Laguerre functions in predictive control, when using Kautz functions to describe the derivative of the control trajectory, we have

$$\dot{u}(\tau) = L_k(\tau)^T \eta = B_k^T(e^{A_k \tau})^T C_k^T \eta. \quad (6.81)$$

For example, with the assumption of a pair of real poles $(-\alpha_1, -\alpha_2)$ and a pair of complex poles $(-\alpha_3 \pm j\beta_3)$, the system matrices to describe a set of Kautz functions are

$$A_k = \begin{bmatrix} -\alpha_1 & 0 & 0 & 0 & 0 \\ -2\alpha_1 & -\alpha_2 & 0 & 0 & 0 \\ -2\alpha_1 & -\alpha_2 & -p_3 & 0 & 0 \\ -2\alpha_1 & -\alpha_2 & \gamma_3 - p_3 & -\bar{p}_3 & 0 \\ -2\alpha_1 & -\alpha_2 & \gamma_3 - p_3 & -\gamma_3 - \bar{p}_3 & -\gamma_3 \end{bmatrix}; \quad B_k^T = [1 \ 1 \ 1 \ 1 \ 1]$$

$$C_k = \begin{bmatrix} \sqrt{2\alpha_1} & 0 & 0 & 0 & 0 \\ 0 & \sqrt{2\alpha_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{2\alpha_3} & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2\alpha_3} \end{bmatrix},$$

where $p_3 = -\alpha_3 + j\beta_3$, $\bar{p}_3 = -\alpha_3 - j\beta_3$, $\gamma_3 = \sqrt{\alpha_3^2 + \beta_3^2}$; $\alpha_1, \alpha_2, \alpha_3 > 0$. Note that the A_k matrix is a lower triangular matrix, which leads to a simplified solution in the design.

The prediction of the state at the future time τ from time t_i is

$$x(t_i + \tau | t_i) = e^{A\tau} x(t_i) + \int_0^\tau e^{A(\tau-\gamma)} B u(\gamma) d\gamma$$

$$= e^{A\tau} x(t_i) + \int_0^\tau e^{A(\tau-\gamma)} B B_k^T (e^{A_k \gamma})^T d\gamma C_k^T \eta.$$

Similar to the situation when Laguerre functions are used, we let

$$\phi(\tau)^T = \int_0^\tau e^{A(\tau-\gamma)} B B_k^T (e^{A_k \gamma})^T d\gamma C_k^T. \quad (6.82)$$

Furthermore, we recognize that the case is identical to the case presented in the Laguerre functions by defining

$$\hat{\phi}(\tau)^T = \int_0^\tau e^{A(\tau-\gamma)} BB_k^T (e^{A_k\gamma})^T d\gamma. \quad (6.83)$$

For a given τ , the matrix $\hat{\phi}(\tau)^T$ can be found analytically through the algebraic equation

$$A\hat{\phi}(\tau)^T - \hat{\phi}(\tau)^T A_k^T = -BB_k^T (e^{A_k\tau})^T + e^{A\tau} BB_k^T \quad (6.84)$$

$$\hat{\phi}(\tau)^T = \hat{\phi}(\tau)^T C_k^T. \quad (6.85)$$

Since A_k is a lower triangular matrix, it is straightforward to solve this matrix algebraic equation, as demonstrated in the early part of this chapter (see Section 6.3). The rest of the solution follows exactly the same procedure used in the Laguerre function case to solve the multi-input and multi-output case by decomposing the multi-input system as single-input systems.

Kautz functions have more flexibility in their pole locations. We utilize this flexibility in the situation where the desired closed-loop poles are exactly known for a given cost function characterized by the pair of Q and R matrices. For instance, assuming that the state variable vector x has dimension n , with the optimal solution from LQR, the set of closed-loop eigenvalues are $\lambda_1, \lambda_2, \dots, \lambda_n$, then by choosing the poles of the Kautz functions to be identical to $\lambda_1, \lambda_2, \dots, \lambda_n$, in theory, the solution of the continuous-time predictive control for a sufficiently large prediction horizon would be identical to the LQR solution without error. In practice, because the algorithm is not numerically stable for a large prediction horizon, the numerical instability causes some errors, but the numerical errors can be removed using exponential data weighting (see Chapter 8). The scenario here is that when the number of real and complex Kautz functions is equal to n and the pole locations of Kautz functions are selected to be equal to the eigenvalues of LQR system, there is no approximation error between the structures of the control trajectories of the predictive control system and the LQR system.

Example 6.6. Suppose that an augmented state-space model for a continuous-time system is described by

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (6.86)$$

$$y(t) = Cx(t), \quad (6.87)$$

where the matrices (A, B, C) are

$$A = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 \\ -2 & -3 & 0 & 0 & 0 \\ -1 & -2 & -5 & 0 & 0 \\ 1 & 2 & 3 & -5 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}; B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}; C = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Choosing $Q = C^T C$ and $R = \begin{bmatrix} 0.2000 & 0 \\ 0 & 0.2000 \end{bmatrix}$, design the LQR control system first. Using LQR's optimal poles as the poles of Kautz functions, calculate the feedback control gain matrix K_{mpc} via minimisation of the predictive control cost function J , where

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(t_i),$$

compare the predictive control solution with the LQR solution in terms of feedback gain and closed-loop pole locations. The prediction horizon $T_p = 8$.

Solution. We first use MATLAB function for LQR design by calling

```
[K_lqr, S, E] = lqr(A, B, Q, R);
```

The eigenvalues of the LQR system contained in the vector E , having the values

$$E = [-5.5081 - 3.9681 \pm j0.2918 - 1.1066 \pm j1.0251 - 1.4113],$$

and the feedback control gain matrix is

$$K_{lqr} = \begin{bmatrix} 1.6234 & -0.3840 & 0.0028 & 0.0073 & 2.2356 & 0.0478 \\ -0.5812 & 0.8302 & 0.2074 & 0.4050 & -0.0478 & 2.2356 \end{bmatrix}. \quad (6.88)$$

There are two pairs of complex poles and two real poles. Selecting the poles of Kautz functions to be identical to E , we construct the state-space descriptions A_k, B_k, C_k for the Kautz functions using the form proposed in Chapter 5.

The solution of the continuous-time MPC problem leads to MPC feedback control gain,

$$K_{mpc} = \begin{bmatrix} 1.9444 & -0.4572 & 0.0070 & 0.0165 & 2.5149 & 0.0973 \\ -0.7961 & 0.8844 & 0.2086 & 0.4032 & -0.2673 & 2.2225 \end{bmatrix}. \quad (6.89)$$

The eigenvalues of the closed-loop predictive control system are -5.5081 , $-3.9703 \pm j0.2865$, $-1.2938 \pm j0.9376$, -1.4113 . The results show that the solution using Kautz functions is very close indeed to the LQR solution. The discrepancies come from using the finite horizon $T_p = 8$ that is different from the infinite horizon solution obtained from the steady-state Riccati equation.

Before closing this section, there are a few comments that are related to the choice between Laguerre and Kautz functions. The justification we often have for using Kautz functions is that they allow complex poles to be used in their structures. Thus, they are more effective for modelling a signal that is oscillatory in nature. Namely, it requires more terms to approximate an oscillatory signal using a set of Laguerre functions than using a set of Kautz functions, provided that the set of poles in the Kautz functions is correctly selected. The correct selection of poles in the Kautz functions is based on the

a priori knowledge from the LQR solution of the original problem. On the other hand, it is fair to say that the complex poles lead to a more complicated formulation of the state-space description of the Kautz functions (A_k, B_k, C_k), although they are computed systematically.

The key advantage of using Laguerre functions lies in the simplicity of the algorithm. Without *a priori* knowledge of the underlying optimal LQR system, the pair of parameters (p, N) can be treated as tuning parameters for closed-loop performance. Because the eigenvalues of the closed-loop system are never intentionally designed to be severely under-damped, the task of modelling the closed-loop control trajectory is adequate for a set of Laguerre functions that have identical real poles. The appropriate choice of (Q, R) matrices will ensure that the closed-loop eigenvalues are suitably damped. Based on the above reasons, the orthonormal functions used in the design in both continuous-time and discrete-time cases are mainly Laguerre functions in most practical applications.

6.8 Summary

This chapter has discussed continuous-time model predictive control using orthonormal basis functions. This chapter is the counterpart of the discrete-time model predictive control, introduced in Chapter 3. The central idea is to model the derivative of the control signal, whose integral squared value is bounded, using a set of orthonormal basis functions. As a result, the continuous-time model predictive control system is designed using a similar framework to the discrete-time counterpart.

However, the continuous-time predictive control is designed based on a continuous-time model, which is independent of the sampling interval at the design stage, even though the implementation is carried out in a digital environment. Because the sampling interval is used in the implementation stage, it permits irregular sampling, and could offer more flexibility and better performance with a fast sampling rate.

Similar to discrete-time MPC, the augmented model is embedded with integrator(s). Thus, the prediction horizon is limited to a finite value, and is used as a tuning parameter in the design. Also, the algorithms suffer from a numerical problem when the prediction horizon is large. This problem will be discussed and overcome in Chapter 8.

Gawthrop and colleagues proposed pole-placement predictive control systems (Gawthrop, 2000, Gawthrop and Ronco, 2002), where a set of exponential functions, chosen corresponding to the underlying poles of the LQR system, is used to describe the control signal $u(t)$. The distinguishing point between Gawthrop's approach and the approach in this section is that the Kautz functions are exponential functions with orthonormality, and the Kautz filters have the same poles of the LQR system, but their zeros are determined through the

orthonormal properties. It is the orthonormal properties of the Kautz functions that lead to the guaranteed convergence of predictive control trajectory to the underlying optimal control trajectory, as the number of terms increases.

The initial idea of using continuous-time Laguerre functions in predictive control was published in Wang (2000, 2001b).

Problems

6.1. Consider the augmented model for a continuous-time system given by

$$\dot{x}(t) = \begin{bmatrix} -2 & 0 \\ 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 2 \\ 0 \end{bmatrix} \dot{u}(t); \quad y(t) = [0 1] x(t). \quad (6.90)$$

Assume that the optimal control signal is to minimize the cost function

$$J = \int_0^{\infty} (x(t)^T Q x(t) + \dot{u}(t)^T R \dot{u}(t)) dt, \quad (6.91)$$

where $Q = C^T C$ and $R = 1$.

1. Using the MATLAB ‘lqr’ function, find the feedback control gain matrix K_{lqr} .
2. With the initial condition $x(0) = [2 \ 3]^T$, compute the optimal control $\dot{u}(t) = -K_{lqr} e^{A_{cl} t}$ where $A_{cl} = A - BK_{lqr}$ is the closed-loop system matrix, and t is specified as $t = 0, \Delta t, 2\Delta t, \dots, T_m$ with $\Delta t = 0.01$ and $T_m = 3$.
3. Find the approximation of the control trajectory using a set of Laguerre functions. Suppose that $p = 1$. What is the number of terms, N , required so that the error equation below is satisfied?

$$\int_0^{T_m} (\dot{u}(t) - \sum_i c_i l_i)^2 dt < 0.01. \quad (6.92)$$

4. Change the initial condition of the state variables to $x(0) = [-1 \ -6]^T$, and repeat the same procedure. Observe how the coefficients c_1, c_2, \dots, c_N have changed due to the change of the initial condition.

6.2. Assuming that $R = \beta$, the closed-loop poles of the optimal control that minimizes the cost function (Kailath, 1980)

$$J = \int_0^{\infty} (y(t)^T y(t) + \dot{u}(t)^T R \dot{u}(t)) dt, \quad (6.93)$$

are the roots of the left-half-plane of the solution

$$D(s^2) = 1 + \frac{1}{\beta^2} G(s) G(-s) = 0.$$

In the modelling of optimal control trajectory, we can use this knowledge to find the optimal p in the Laguerre network. An approximate approach is to take the real part of the dominant pole of the LQR system as the $-p$ value. Find the parameter p for Problem 6.1. What is the number of terms, N , required to achieve the accuracy specified in (6.92)?

6.3. A continuous-time system has the transfer function

$$G(s) = \frac{(1 - 3s)}{(s^2 + 2 \times 0.2s + 1)(1 + 3s)},$$

where part of the transfer function $\frac{1-3s}{1+3s}$ is due to the approximation of the time delay factor e^{-3s} .

1. Find the augmented state-space model that will be used as the basis for the design of continuous-time MPC.
2. Choosing $Q = C^T C$, $R = 0.1$, $N = 5$, $p = 1$, and $T_p = 6$, calculate the Ω and Ψ matrices in the corresponding cost function

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(t_i),$$

where the computational interval used in obtaining Ω and Ψ is selected to be a small fraction of $\frac{1}{p}$.

3. With given initial condition $x(t_i) = [1 \ 0 \ -1 \ 2]^T$ and set-point signal $r(t_i) = 1$ at time $t_i = 10$, compute the optimal state response for $t = t_i, t_i + \Delta t, t_i + 2\Delta t, \dots, t_i + T_p$, where $\Delta t = 0.01$. Present the results graphically.
4. What are your observations of these responses?

6.4. Continue from Problem 6.3. Experiment with the tuning parameters. The experiments should include the following combinations.

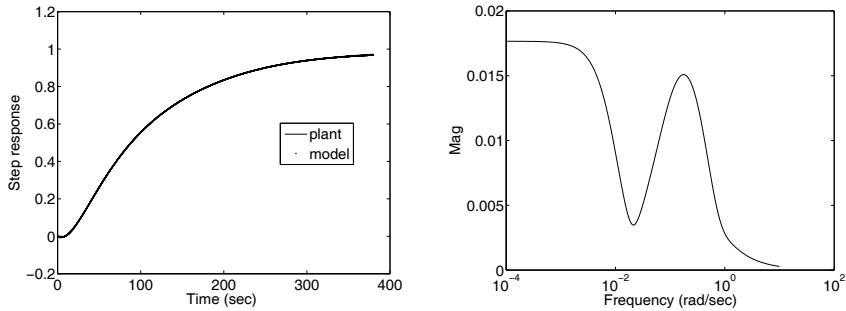
1. $p = 1$, $R = 1$, $N = 2$, the other parameters being unchanged.
2. $p = 1$, $R = 0.01$, $N = 8$, the other parameters being unchanged.
3. $N = 2$, $R = 0.5$, vary p between 0.5 and 5.

Compare the state variables in the above cases with the results obtained from Problem 6.3. What are your observations on R and N ?

6.5. In the context of state-space design, when using Laguerre models, the state variable information can be constructed directly from plant input variables. Therefore, an observer is not required in the control system implementation.

A continuous-time plant is described by the transfer function model

$$G(s) = \frac{0.001(s - 2)(s - 0.5)}{(s + 2)(s + 1)^4(s + 0.5)(s + 0.1)(s + 0.001)}. \quad (6.94)$$



(a) Comparison between the step responses (b) The magnitude of frequency error ($|G(jw) - G_m(jw)|$)

Fig. 6.6. Illustration of the accuracy of the Laguerre model

By optimizing the scaling factor p , where the optimal p is found to be 0.0222, this continuous-time transfer function model can be quite accurately approximated with a fourth-order Laguerre model

$$G_m(s) = c_1 L_1(s) + c_2 L_2(s) + c_3 L_3(s) + c_4 L_4(s),$$

where the coefficients are

$$c_1 = 0.0439, \quad c_2 = -0.0500, \quad c_3 = 0.0012, \quad c_4 = -0.0084,$$

and the Laguerre filters are

$$\begin{aligned} L_1(s) &= \frac{\sqrt{2p}}{s+p}; \quad L_2(s) = \frac{\sqrt{2p}(s-p)}{(s+p)^2}; \\ L_3(s) &= \frac{\sqrt{2p}(s-p)^2}{(s+p)^3}; \quad L_4(s) = \frac{\sqrt{2p}(s-p)^3}{(s+p)^4}. \end{aligned}$$

Figure 6.6 shows the comparison results of the step and frequency responses. It is seen from this figure that the Laguerre model is indeed an excellent approximation of the original transfer function model.

1. Show that the matrices (A_m, B_m, C_m) for the Laguerre state-space model are

$$A_m = \begin{bmatrix} -p & 0 & 0 & 0 \\ -2p & -p & 0 & 0 \\ -2p & -2p & -p & 0 \\ -2p & -2p & -2p & -p \end{bmatrix}, \quad B_m = \sqrt{2p} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

$$\text{and } C_m = [c_1 \ c_2 \ c_3 \ c_4].$$

2. Design a predictive control system based on this state-space model that will follow a constant input signal without steady-state error. The design specifications are: $N = 3$, $p = 0.1$, $Q = C^T C$, $R = 1$, $T_p = 300$.

3. What is the feedback control gain matrix K_{mpc} ? Where are the closed-loop poles allocated?
4. Because the state variables can be constructed through the derivative of the control signal, there is no need for the use of an observer in the implementation of the predictive control system. Simulate the nominal closed-loop system with a unit step reference signal and sampling interval $\Delta t = 0.01$. Also, perform the same closed-loop simulation using the original order model, and compare the closed-loop response with the nominal closed-loop response.

6.6. A mechanical system is described by a second-order transfer function model

$$G(s) = \frac{s - 1}{s^2 + 0.02s + 1}.$$

1. Use the MATLAB function ‘tf2ss’ to obtain the state-space representation for this transfer function model, and find the augmented state-space model.
2. Design a predictive control system that will have sufficient damping and follow a step set-point change. Use the left-half-plane roots of $D(s^2) = 1 + \frac{1}{\beta^2} \frac{G(s)G(-s)}{-s^2}$ to find the value of β that will produce the desired closed-loop poles that you prefer.
3. This set of closed-loop poles are utilized in the Kautz functions for design of continuous-time MPC. The prediction horizon T_p is chosen to be 6. Verify that the closed-loop poles of the predictive control system are close to the desired closed-loop poles.
4. Implement the predictive control system with an observer, where the observer gain K_{ob} is selected so that the closed-loop observer system has poles twice the magnitude of the control loop poles.

Continuous-time MPC with Constraints

7.1 Introduction

This chapter discusses continuous-time model predictive control with constraints. Section 7.2 formulates the constraints for the continuous-time predictive control system, including the cases of set-point following and disturbance rejection of constant signals. Section 7.3 presents the numerical solution of the constrained control problem with an example, where Hildreth's quadratic programming procedure is employed. Because of the nature of the continuous-time formulation such as fast sampling, there might be computational delays when the quadratic programming procedure is used in the solution of the real-time optimization problem. Section 7.4 discusses the real-time implementation of continuous-time model predictive control in the presence of constraints.

7.2 Formulation of the Constraints

7.2.1 Frequently Used Constraints

There are three types of constraints frequently encountered in continuous-time applications. The first two types deal with constraints imposed on the manipulated variables, and the third type of constraint deals with output or state variable constraints.

Constraints on the Manipulated Variable Rate of Change

In the context of continuous-time control, the rate of change on the control signal is given by the limit:

$$\lim_{\Delta t \rightarrow 0} \frac{\Delta u(t_i) - \Delta u(t_i - \Delta t)}{\Delta t} = \dot{u}(t) |_{t=t_i} . \quad (7.1)$$

We often use an approximation of the derivative for a given small Δt to calculate the upper and lower limits of the derivative. Assume that for a single-input system, the upper limit is du^{max} and the lower limit is du^{min} . Then, constraints are specified as

$$du^{min} \leq \dot{u}(t) \leq du^{max}. \quad (7.2)$$

Note that we use \leq , not $<$, which includes the case of $=$. This is because the optimal solution involves the active constraints, and they are the equality ones. If we have more than one input, say m inputs, then we can specify the constraints for each input independently. In the multi-input case, suppose that the constraints are given for the upper limit as

$$[du_1^{max} \ du_2^{max} \ \dots \ du_m^{max}],$$

and lower limit as

$$[du_1^{min} \ du_2^{min} \ \dots \ du_m^{min}].$$

We can specify each variable with rate of change, as

$$\begin{aligned} du_1^{min} &\leq \dot{u}_1(t) \leq du_1^{max} \\ du_2^{min} &\leq \dot{u}_2(t) \leq du_2^{max} \\ &\vdots \\ du_m^{min} &\leq \dot{u}_m(t) \leq du_m^{max}. \end{aligned} \quad (7.3)$$

The derivative constraints can be used to impose directions of movement on the manipulated variables. For instance, if $u_1(t)$ can only increase, and not decrease, then we select $0 \leq \dot{u}_1(t) \leq du_1^{max}$.

Constraints on the Amplitude of the Manipulated Variable

Suppose that the constraints are given for the upper limit of the control signal as

$$[u_1^{max} \ u_2^{max} \ \dots \ u_m^{max}],$$

and lower limit as

$$[u_1^{min} \ u_2^{min} \ \dots \ u_m^{min}].$$

Then, we specify the amplitude of each control signal to satisfy the constraints:

$$\begin{aligned} u_1^{min} &\leq u_1(t) \leq u_1^{max} \\ u_2^{min} &\leq u_2(t) \leq u_2^{max} \\ &\vdots \\ u_m^{min} &\leq u_m(t) \leq u_m^{max}. \end{aligned} \quad (7.4)$$

Output Constraints

We can also specify the operating range for the plant output. For instance, supposing that the output $y(t)$ has an upper limit y^{max} and a lower limit y^{min} , then the output constraints are specified as

$$y^{min} \leq y(t) \leq y^{max}. \quad (7.5)$$

Output constraints are often used in the regulation case where disturbance rejection is the primary focus for control. However, because output constraints could cause a predictive control system to become unstable, we need to be cautious when implementing output constraints. So they are often implemented as ‘soft’ constraints and a slack variable $s_v > 0$ is added to the constraints, forming

$$y^{min} - s_v \leq y(t) \leq y^{max} + s_v. \quad (7.6)$$

This means that they will not become active if the slack variable s_v is chosen large enough.

There are several reasons why we use a slack variable to form ‘soft’ constraints for outputs. One is that the output constraints often cause the problem of conflict constraints in the situation where the input constraints become activated. Also, when the predictive control system tries to alter the behaviour of a plant output, severe nonlinearity appears in the control law, which may result in closed-loop system oscillation, and system instability. We have observed this in the discrete-time case when output constraints are enforced (see Section 2.5.4 and Section 3.8.1). Similarly, this can occur in continuous-time control. When this happens, we relax the output constraints by selecting a larger slack variable s_v to resolve the problem so that the active constraints do not become active. Another reason is that the constraints are imposed based on the model and in the case where there is a model-plant mismatch, the prediction of the output may not be accurate. The constraints are not meaningful unless the model used for prediction is reasonably accurate.

Similarly, we can impose constraints on the state variables if they are measured or estimated. They also need to be in the form of ‘soft’ constraints for the same reasons as the output case.

7.2.2 Constraints as Part of the Optimal Solution

Having formulated the constraints as part of design requirements, the next step is to translate them into linear inequalities, and relate them to the original model predictive control problem. The key here is to parameterize the constrained variables using the same orthonormal basis functions as the ones used in the design of predictive control. Subsequently, we represent the constraints in terms of the parameter vector η . Since the predictive control problem is formulated and solved in the framework of receding horizon control, the

constraints are taken into consideration frame-by-frame for each moving horizon window. This allows us to vary the constraints at the beginning of each frame and also gives us the means to solve the constrained control problem numerically.

Constraints on the Derivative of Control

If we want to impose the constraints on the derivative of the control signal $\dot{u}(t)$ at time t_i , the constraints are expressed as

$$du^{min} \leq \dot{u}(t) \leq du^{max},$$

where du^{min} and du^{max} are the minimum and maximum limits of the derivative at time $t = t_i$. From the time instance t_i , the predictive control scheme looks into the future, and we need to express the future of the derivative of the control signal in terms of the Laguerre coefficient vector η . Differing from the discrete-time case, the future control trajectory in the continuous-time case is represented by continuous-time Laguerre functions. Thus, in order to obtain a finite set of linear inequalities for the constraints, the future time within the optimization window is discretized to obtain the time intervals $0, \tau_1, \tau_2, \dots$ for which we wish to impose the constraints. The parameters τ_1, τ_2, \dots , may not be related to the sampling interval Δt used for the implementation of the continuous-time predictive control, however, the first sample $\tau_0 = 0$ is always imposed to ensure that the constraints are satisfied when the receding horizon principle is applied. The implementation of the constraints on the first sample is performed as

$$-L(0)^T \eta \leq -du^{min} \quad (7.7)$$

$$L(0)^T \eta \leq du^{max}. \quad (7.8)$$

At an arbitrary future time τ_i , the derivative of the control signal is expressed as

$$du^{min} \leq \dot{u}(\tau_i) \leq du^{max}. \quad (7.9)$$

Note that

$$\dot{u}(\tau_i) = L(\tau_i)^T \eta,$$

where $L(\tau_i) = [l_1(\tau_i) \ l_2(\tau_i) \ \dots \ l_N(\tau_i)]^T$ is the vector of the Laguerre functions and η is the parameter vector. Thus, the inequalities for the constraints at the time τ_i are

$$-L(\tau_i)^T \eta \leq -du^{min} \quad (7.10)$$

$$L(\tau_i)^T \eta \leq du^{max}. \quad (7.11)$$

The inequalities are readily extended to a multi-input system, as illustrated by the following example.

Example 7.1. Consider a continuous-time system with two inputs, u_1 and u_2 . The constraints for the derivatives of the control signals are specified as

$$du_1^{min} \leq \dot{u}_1(t) \leq du_1^{max}; \quad du_2^{min} \leq \dot{u}_2(t) \leq du_2^{max}, \quad (7.12)$$

where $du_1^{min} = 0.5$, $du_1^{max} = 0.8$, $du_2^{min} = -0.3$, $du_2^{max} = 1$. Within the optimization window, the constraints are imposed on the first sample $\tau_0 = 0$ and $\tau_1 = 0.2$. Assuming $p_1 = 0.6$ and $p_2 = 1$; $N_1 = N_2 = 2$, formulate the inequalities for imposing the constraints in the solution of continuous-time predictive control.

Solution. In the design of predictive control, the two derivatives of the control are expressed as

$$\dot{u}_1(\tau) = L_1(\tau)^T \eta_1; \quad \dot{u}_2(\tau) = L_2(\tau)^T \eta_2. \quad (7.13)$$

Letting $\eta = [\eta_1^T \eta_2^T]^T$ with η_1 and η_2 having two coefficients each ($N_1 = N_2 = 2$), the inequalities corresponding to the constraints are

$$\begin{bmatrix} L_1(0)^T & o_2^T \\ o_1^T & L_2(0)^T \end{bmatrix} \eta \leq \begin{bmatrix} du_1^{max} \\ du_2^{max} \end{bmatrix} \quad (7.14)$$

$$\begin{bmatrix} L_1(\tau_1)^T & o_2^T \\ o_1^T & L_2(\tau_1)^T \end{bmatrix} \eta \leq \begin{bmatrix} du_1^{max} \\ du_2^{max} \end{bmatrix} \quad (7.15)$$

$$-\begin{bmatrix} L_1(0)^T & o_2^T \\ o_1^T & L_2(0)^T \end{bmatrix} \eta \leq \begin{bmatrix} -du_1^{min} \\ -du_2^{min} \end{bmatrix} \quad (7.16)$$

$$-\begin{bmatrix} L_1(\tau_1)^T & o_2^T \\ o_1^T & L_2(\tau_1)^T \end{bmatrix} \eta \leq \begin{bmatrix} -du_1^{min} \\ -du_2^{min} \end{bmatrix}, \quad (7.17)$$

where o_1 and o_2 are the zero vectors with their dimensions equal to those in $L_1(0)$ and $L_2(0)$, respectively. Numerically, these eight linear inequalities are shown as

$$\begin{bmatrix} 1.0954 & 1.0954 & 0 & 0 \\ 0 & 0 & 1.4142 & 1.4142 \\ 0.9716 & 0.7384 & 0 & 0 \\ 0 & 0 & 1.1579 & 0.6947 \\ -1.0954 & -1.0954 & 0 & 0 \\ 0 & 0 & -1.4142 & -1.4142 \\ -0.9716 & -0.7384 & 0 & 0 \\ 0 & 0 & -1.1579 & -0.6947 \end{bmatrix} \eta \leq \begin{bmatrix} 0.8 \\ 1 \\ 0.8 \\ 1 \\ -0.5 \\ 0.3 \\ -0.5 \\ 0.3 \end{bmatrix}. \quad (7.18)$$

MATLAB tutorial: how to formulate the derivative constraints

Tutorial 7.1. The purpose of this tutorial is to show how to formulate the derivative constraints using MATLAB.

Step by Step

1. Create a new file called *Mder.m*.
2. We only impose constraints on the first sample $\tau_0 = 0$ and a sample at τ in the future time. The program can be easily modified to include a larger number of constraints within the optimization window. The parameters p and N are the parameters for the Laguerre functions; n_{in} is the number of inputs and τ is the future sample of the constraints to be imposed. The first part of the program will formulate the constraints for the initial sample, so $L(0)$ will be generated and used. Because it is a multi-input system, attention is paid to the dimension of the matrix.
3. Enter the following program into the file:

```
function [M_dv,Lzerot]=Mder(p,N,n_in,tau)
N_pa=sum(N);
k0=1;
[A1,LO]=lagc(p(k0),N(k0));
L_t=zeros(n_in,N_pa);
L_t(1,1:N(1))=LO';
cc=N(1);
for k0=2:n_in;
[A1,LO]=lagc(p(k0),N(k0));
L_t(k0,cc+1:cc+N(k0))=LO';
cc=cc+N(k0);
end
```

4. The second part of the program is to formulate the constraints for the second sample at time τ .

5. Continue entering the following program into the file:

```
Lzerot=L_t;
k0=1;
[A1,LO]=lagc(p(k0),N(k0));
L1=expm(A1*tau)*LO;
L_t1=zeros(n_in,N_pa);
L_t1(1,1:N(1))=L1';
cc=N(1);
for k0=2:n_in;
[A1,LO]=lagc(p(k0),N(k0));
L1=expm(A1*tau)*LO;
L_t1(k0,cc+1:cc+N(k0))=L1';
cc=cc+N(k0);
end
M_dv=[L_t1];
```

6. Test the function with the following parameters

```
p=[0.2 1];
```

```
N=[2 3];
n_in=2;
tau=0.3;
```

7. The results are

$$\begin{array}{cccccc} M_{dv} & 0.5956 & 0.5241 & 0 & 0 & 0 \\ & 0 & 0 & 0.7079 & 0.5805 & 0.4645 \\ \\ Lzerot & 0.6325 & 0.6325 & 0 & 0 & 0 \\ & 0 & 0 & 0.7746 & 0.7746 & 0.7746 \end{array}$$

Constraints on the Amplitude of Control

As for the control signal, assuming that Δt is the sampling interval for implementation, the first sample of the control signal at the optimization window, is calculated as

$$u(t_i) = u(t_i - \Delta t) + L(0)^T \eta \Delta t, \quad (7.19)$$

where $L(0)^T \eta$ is the derivative of control at the beginning of the optimization window. This leads to the constraints for the control signal at the first sample time of the window as

$$u^{min} - u(t_i - \Delta t) \leq L(0)^T \eta \Delta t \leq u^{max} - u(t_i - \Delta t), \quad (7.20)$$

which is in agreement with how the control is calculated using the velocity form. At the arbitrary time τ_i , we have

$$u(\tau_i) = u(t_i) + \int_0^{\tau_i} \dot{u}(\gamma) d\gamma \quad (7.21)$$

$$= u(t_i) + \int_0^{\tau_i} L(\gamma)^T \eta d\gamma \quad (7.22)$$

$$= u(t_i) + (L(\tau_i)^T - L(0)^T) A_p^{-T} \eta. \quad (7.23)$$

With the information of $u(t_i - \Delta t)$, the future control signal at time τ_i ($\neq 0$) is expressed as

$$u(\tau_i) = u(t_i - \Delta t) + \overbrace{(L(0)^T \Delta t + L(\tau_i)^T A_p^{-T} - L(0)^T A_p^{-T}) \eta}^{C_u}. \quad (7.24)$$

Therefore, the inequality constraints are formulated as

$$u^{min} - u(t_i - \Delta t) \leq C_u \eta \leq u^{max} - u(t_i - \Delta t), \quad (7.25)$$

where u^{min} and u^{max} are the lower and upper limits of control signal u .

Note that the choice of the set of future time instants plays an important role in the numerical solution of the constrained control problem. In the

continuous-time case, the choice of $\tau_1, \tau_2, \dots, \tau_m$ needs to be considered carefully. If they are selected too close to each other, then the constraints will be approximately equal to each other, which leads to redundancy among them. An appropriate choice will reduce the number of constraints.

Tutorial 7.2. *This tutorial shows how to produce the data matrix C_u for a multi-input system.*

Step by Step

1. Create a new file called *Mucon.m*.
2. We only impose constraints on the first sample $\tau_0 = 0$ and a sample at τ in the future time. The program can be easily modified to include a larger number of constraints within the optimization window. The parameters p and N are the parameters for the Laguerre functions; n_{in} is the number of inputs and τ is the future time of the constraints to be imposed. The first part of the program will formulate the constraints for the initial sample, so $L(0)\Delta t$ will be generated and used. Because it is a multi-input system, attention is paid to the dimension of the matrix.
3. Enter the following program into the file:

```
function [Mu,Mu1]=Mucon(p,N,n_in,delta_t,tau)
%function for generating matrix M for
%the constraints on the control signal
%constraints are imposed on the zero time and tau time
%delta-t is the sampling interval
%Mu is for constraints to be imposed on the zero sample
%Mu1 is for constraints to be imposed on tau time
N_pa=sum(N);
k0=1;
[A1,L0]=lagc(p(k0),N(k0));
L_t=zeros(n_in,N_pa);
L_t(1,1:N(1))=L0';
cc=N(1);
for k0=2:n_in;
    [A1,L0]=lagc(p(k0),N(k0));
    L_t(k0,cc+1:cc+N(k0))=L0';
    cc=cc+N(k0);
end
% constraints on second sample
k0=1;
[A1,L0]=lagc(p(k0),N(k0));
L1=expm(A1*tau)*L0;
L_t1=zeros(n_in,N_pa);
L_t1(1,1:N(1))=(L1'-L0')*inv(A1')+L0'*delta_t;
cc=N(1);
```

```

for k0=2:n_in;
[A1,L0]=lagc(p(k0),N(k0));
L1=expm(A1*tau)*L0;
L_t1(k0,cc+1:cc+N(k0))=(L1'-L0')*inv(A1')+L0'*delta_t;
cc=cc+N(k0);
end
Mu=[L_t*delta_t];
Mu1=[L_t1];

```

4. Test the function using the following parameters:

```

p=[1 2];
N=[3 2];
n_in=2;
tau=1;
delta_t=0.1;

```

5. The results are

$$\begin{array}{cccccc}
\text{Mu} = & 0.1414 & 0.1414 & 0.1414 & 0 & 0 \\
& 0 & 0 & 0 & 0.2000 & 0.2000 \\
\\
\text{Mu1} = & 1.0354 & 0.2880 & -0.0051 & 0 & 0 \\
& 0 & 0 & 0 & 1.0647 & -0.1233
\end{array}$$

7.3 Numerical Solutions for the Constrained Control Problem

Now, the predictive control problem with hard constraints imposed in the design becomes the problem of finding the optimal solution of the quadratic cost function:

$$\begin{aligned}
J = & \eta^T \Omega \eta + 2\eta^T \Psi x(t_i) \\
& + x(t_i)^T \int_0^{T_p} e^{A^T \tau} Q e^{A \tau} d\tau x(t_i),
\end{aligned} \tag{7.26}$$

where Ω and Ψ are given as

$$\Omega = \int_0^{T_p} \phi(\tau) Q \phi(\tau)^T d\tau + R_L \tag{7.27}$$

$$\Psi = \int_0^{T_p} \phi(\tau) Q e^{A \tau} d\tau \tag{7.28}$$

$$\phi(\tau)^T = \int_0^\tau e^{A(\tau-\gamma)} B L(\gamma)^T d\gamma, \tag{7.29}$$

subject to the linear inequality constraints that are formed from the previous analysis. When set-point following is required in the predictive control, if it

is a piece-wise constant signal, the last block element corresponding to the output y will be the error signal between the actual and desired signals of the output.

Finding the numerical solution to the continuous-time predictive control with constraints is concerned with the problems of constrained minimization where the constraint functions are linear and the objective function is a positive definite quadratic function. Similar to the discrete-time case, because the constraint functions are expressed in the form of linear inequalities, in general, the solutions involve quadratic programming procedures as outlined in Chapter 2.

Tutorial 7.3. *In this tutorial, we will produce a MATLAB function that performs constrained control for a continuous-time system. The program is written for a MIMO system, and we only impose constraints on the first sample of the control signals (both derivative and amplitude). sp is the set-point signal, which has the same number of rows as the output and number of columns greater than or equal to the simulation time N_{sim} . The plant model is described by (A_p, B_p, C_p) , and the augmented model is described by (A, B, C) . $Lzerot$ is the data matrix for reconstructing the derivative of the control from Laguerre functions. M is the data matrix for the inequality constraints ($M\eta \leq \gamma$) where η is the Laguerre parameter vector. h is the sampling interval. The initial conditions of the plant are specified by the data vectors xm, u, y .*

Step by Step

1. Create a new file called `cssimucon.m`. We will set the initial conditions and the data vector γ . Enter the following program into the file:

```
function [u1,y1,udot1,t]=
cssimucon(xm,u,y,sp,Ap,Bp,Cp,A,B,C,N_sim,Omega,Psi,
K_ob,Lzerot,h,M,u_max,u_min,Deltau_max,Deltau_min);
up=u;
gamma=[(u_max-up);(-u_min+up);Deltau_max;-Deltau_min];
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
X_hat=zeros(n1+m1,1);
```

2. In order to incorporate the set-point signal in the simulation, we define a vector Xsp , which has the same dimension as the observed state with all zero elements except the last $m1$ rows as the set-point signal. The state feedback variable Xf is defined and the constrained control problem is solved using the MATLAB function `QPhild.m` (see Section 2.4.4 in Chapter 2). With the optimized $udot$ (\dot{u}), the control and observer state are updated. Continue entering the following program into the file:

```
for kk=1:N_sim;
Xsp=[zeros(n1,1);sp(:,kk)];
```

```

Xf=X_hat-Xsp;
eta=QPhild(Omega,Psi*Xf,M,gamma);
udot=Lzerot*eta;
u=u+udot*h;
udot1(1:n_in,kk)=udot;
u1(1:n_in,kk)=u;
y1(1:m1,kk)=y;
X_hat=X_hat+(A*X_hat+K_obs*(y-C*X_hat))*h+B*udot*h;

```

3. We update the plant state and output in the simulation. (If the predictive control were implemented on a real plant, then the control signal would be sent to the actuator and the output y would be the signal sensed in the plant operation.) Continue entering the following program into the file:

```

xm=xm+(Ap*xm+Bp*u)*h;
y=Cp*xm;
up=u;
gamma=[(u_max-up);(-u_min+up);Deltau_max;-Deltau_min];
end
t=0:h:(N_sim-1)*h;

```

4. Test your program using the data from Tutorial 7.4.

Tutorial 7.4. In this tutorial, we will present a case study of continuous-time model predictive control with constraints based on an industrial process.

A sugar mill model was presented in Goodwin et al. (2000) for a case study of control system design. In the study, a single stage of a milling train was described by the following continuous-time transfer function model:

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}, \quad (7.30)$$

where the output y_1 is the mill torque, and y_2 is the buffer chute height; the input u_1 is the flap position and input u_2 is the turbine speed set-point. The four transfer functions are given as

$$g_{11}(s) = \frac{-5}{25s + 1}; \quad g_{12}(s) = \frac{s^2 - 0.005s - 0.005}{s(s + 1)};$$

$$g_{21}(s) = \frac{1}{25s + 1}; \quad g_{22}(s) = \frac{-0.0023}{s}.$$

This process has significant multivariable interaction, nonminimum-phase behaviour. Also, the plant contains integrators.

The design specifications for the model predictive control are $Q = C^T C$, $R = I$, $T_p = 100$, $N_1 = N_2 = 6$, $p_1 = p_2 = 0.6$; and the observer design specifications are $Q_1 = I$, and $R_1 = 0.2I$. A sampling interval $\Delta t = 0.03$ (sec) is suggested for this continuous-time system simulation.

The constraints are imposed on the derivatives and amplitudes of both u_1 and u_2 , and we only implement them on the first sample of the signals. The simulation scenario assumes that with zero initial conditions, the output y_1 performs a positive unit set-point change, and output y_2 performs a negative unit set-point change, while the constraints are specified as, $-1 \leq u_1 \leq 0$; $-3 \leq u_2 \leq 3$; $-0.4 \leq \dot{u}_1 \leq 0.4$; $-0.4 \leq \dot{u}_2 \leq 0.4$.

Step by Step

- When the turbine speed set-point u_2 changes, the effect on the mill torque is almost instantaneous, and this is reflected on the model by having $g_{12}(s)$ as a proper transfer function (both numerator and denominator are second-order s -polynomials). The structure of this proper transfer function causes a problem in the design of predictive control, as we have assumed that there is no direct connection between the input and output to allow the computation of receding horizon control law. However, by adding a filter with a very small time constant to $g_{12}(s)$, we obtain the modified transfer function as:

$$g_{12}(s) = \frac{s^2 - 0.005s - 0.005}{s(s+1)(0.1s+1)}.$$

The modified $g_{12}(s)$ is a strictly proper transfer function, and there is no direct link between u_2 and y_1 .

- Create a new MATLAB program called *sugarmill.m*. We will first define the plant and create the augmented state-space model. Enter the following program into the file:

```

num11=-1;
den11=[25 1];
num12=[1 -0.005 -0.005];
den12=conv([1 0],[1 1]);
den12=conv(den12,[0.1 1]);
num21=1;
den21=[25 1];
num22=-0.0023;
den22=[1 0];
Gs=tf({num11 num12; num21 num22},
{den11 den12; den21 den22});
Gs1=ss(Gs,'min');
[Ap,Bp,Cp,Dp]=ssdata(Gs1);
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
A=zeros(n1+m1,n1+m1);
A(1:n1,1:n1)=Ap;
A(n1+1:n1+m1,1:n1)=Cp;
B=zeros(n1+m1,n_in);

```

```
B(1:n1,:)=Bp;
C=zeros(m1,n1+m1);
C(:,n1+1:n1+m1)=eye(m1,m1);
```

3. We enter the design parameters and compute Ω and Ψ . Continue entering the following program into the file:

```
n=n1+m1;
Q=C'*C;
R=1*eye(2,2);
p1=0.6;
p2=0.6;
N1=6;
N2=6;
Tp1=100;
Tp2=100;
p=[p1 p2];
N=[N1 N2];
Tp=[Tp1 Tp2];
[Omega,Psi]=cmpc(A,B,p,N,Tp,Q,R);
```

4. We design the observer. Continue entering the following program into the file:

```
Q1=eye(n,n);
R1=0.2*eye(m1,m1);
K_ob=lqr(A',C',Q1,R1)';
```

5. The initial conditions and the set-point signals are specified. Pay attention to the data structure of the set-point signal. Continue entering the following program into the file:

```
xm=zeros(n1,1);
u=zeros(n_in,1);
y=zeros(m1,1);
h=0.03;
N_sim=8*1400;
sp1=ones(1,N_sim);
sp2=[zeros(1,N_sim/2) -ones(1,N_sim/2)];
sp=[sp1;sp2]; %set-point signal
```

6. Define the constraints by calling ‘Mder.m’ and ‘Mucon.m’. Since we only impose constraints on the first sample, the second sample of constraints is neglected (we choose $\tau = 0.1$ as an arbitrary choice). Continue entering the following program into the file:

```
[Md,Lzerot]=Mder(p,N,n_in,0.1);
[Mu,Mu1]=Mucon(p,N,n_in,h,0.1);
M=[Mu;-Mu;Lzerot;-Lzerot];
```

```

u_max=[0;3];
u_min=[-1;-3];
Deltau_max=[0.4;0.4];
Deltau_min=[-0.4;-0.4];

```

7. The constrained control simulation is performed using the function ‘*cssimucon.m*’. Continue entering the following program into the file:

```

[u1,y1,udot1,t]=cssimucon(xm,u,y,sp,Ap,Bp,Cp,A,B,C,N_sim,
Omega,Psi,K_ob,Lzerot,h,M,u_max,u_min,Deltau_max,
Deltau_min);

```

Figure 7.1 shows the constrained control results for this example. There are eight constraints in the case. So it is difficult to take a guess at the active constraints, and it is necessary to identify the active constraints in the optimization. We have done so by using Hildreth’s quadratic programming algorithm. If we carefully examine the number of iterations required to identify the active constraints in the quadratic programming, then we find that the convergence rate of the Lagrange multipliers is very fast. In fact, it takes about two iterations to achieve the convergence, and this is due to the fact that the active constraints are linearly independent.

It is seen from Figure 7.1 that all the constraints are satisfied. By comparing with the unconstrained case, we see in Figure 7.1 that the response speed of the constrained control system is slightly slower. The Laguerre parameters p_1 , p_2 , N_1 and N_2 could be used as performance related tuning parameters. For instance, in this example, when we choose $p_1 = p_2 = 0.1$ and $N_1 = N_2 = 3$, the closed-loop control results are different from the previous case (see Figure 7.2). In particular, the overshoot and undershoot in y_1 are reduced, and also the constraints on the control signal u_2 become active in shorter time intervals.

7.4 Real-time Implementation of Continuous-time MPC

The essence of continuous-time model predictive control is to minimize the cost function:

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(t_i) + \text{constant},$$

subject to the set of inequality constraints:

$$M\eta \leq \gamma.$$

This formulation in the continuous-time case is fundamentally identical to the one in the discrete-time case. One of the key advantages in using continuous-time predictive control instead of discrete-time predictive control is that the design model and the algorithm are robust in a fast sampling environment. The discrete-time models and predictive control algorithms could encounter

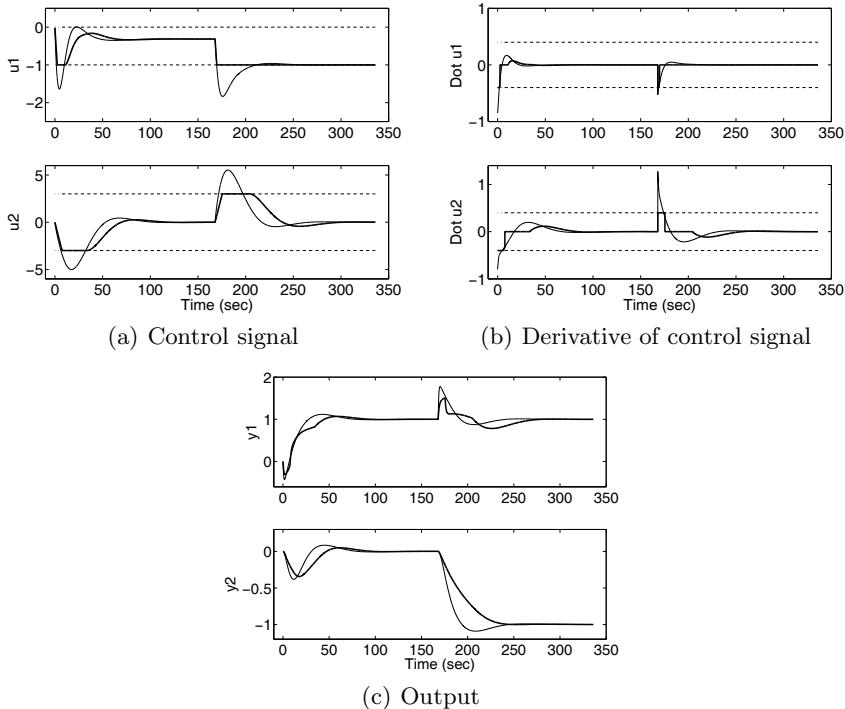


Fig. 7.1. Comparison of CMPC with and without constraints. Key: solid-line, without constraints; darker-solid-line, with constraints ($-1 \leq u_1 \leq 0$; $-3 \leq u_2 \leq 3$; $-0.4 \leq \dot{u}_1$; $-0.4 \leq \dot{u}_2 \leq 4$)

problems when the system is under a fast sampling condition. Another advantage is that because the design is performed in the continuous-time domain, the discretization is carried out in the implementation stage and the framework permits the control of an irregular sampled-data system.

Without imposing constraints, the computational requirement of minimizing the objective function J is negligible because the analytical expression is presented for the optimal solution with Ω and Ψ computed off-line. Therefore, implementing a continuous-time predictive control without constraints is not a challenging task. Also, if it is a single-input and single-output system, then the closed-form solution of the constrained control problem, introduced for discrete-time systems in Chapter 2, can be adapted to the applications in continuous-time. However, when introducing constraints for a multi-input and multi-output system, it is often necessary to use quadratic programming procedures to find the active constraints because the violated constraints may not be the active constraints. An optimal combination of the input signals needs to be found through the iterative search procedure (see Chapter 2). Therefore, there is a larger computational demand for finding the optimal

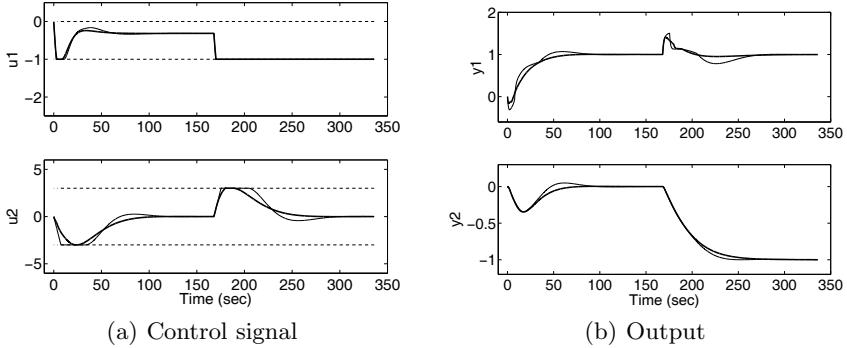


Fig. 7.2. Comparison of CMPC with different performance parameters. Key: solid-line ($p_1 = p_2 = 0.6, N_1 = N_2 = 6$); darker-solid-line ($p_1 = p_2 = 0.1, N_1 = N_2 = 3$); ($-1 \leq u_1 \leq 0; -3 \leq u_2 \leq 3; -0.4 \leq \dot{u}_1; -0.4 \leq \dot{u}_2 \leq 4$)

solution of the constrained control problem for a MIMO system. Since the optimal solution cannot be found instantaneously, a computational delay will occur in the implementation of a continuous-time predictive control. If the computational rate is faster than the sampling rate in implementation, then the computational delay is not an issue. However, if the computational rate is slower than the sampling rate, then the computational delay must be addressed carefully.

Understanding the Problem of Computational Delay

As we understand, at time t_i the receding horizon control is performed within the optimization window $0 \leq \tau \leq T_p$. Closed-loop feedback is introduced through the initial condition of the state variable $x(t_i)$. Or in the general case, $x(t_i)$ is replaced by its estimate $\hat{x}(t_i)$ through an observer. The continuous-time observer is represented by the observer equation

$$\frac{d\hat{x}(t)}{dt} = A\hat{x}(t) + B\dot{u}(t) + K_{ob}(y(t) - C\hat{x}(t)).$$

With approximation for sufficiently small Δt , at time t_i

$$\frac{d\hat{x}(t)}{dt} \approx \frac{\hat{x}(t_i + \Delta t) - \hat{x}(t_i)}{\Delta t}.$$

Therefore, the estimated future state variable $\hat{x}(t_i + \Delta t)$ is calculated based on current values of $\hat{x}(t_i)$, $\dot{u}(t_i)$ and $y(t_i)$, using the formula,

$$\hat{x}(t_i + \Delta t) = (A\hat{x}(t_i) + B\dot{u}(t_i) + K_{ob}(y(t_i) - C\hat{x}(t_i))) \Delta t + \hat{x}(t_i). \quad (7.31)$$

With the estimated future state variable $\hat{x}(t_i + \Delta t)$ and the set-point information $r(t_i + \Delta t)$ as the new initial condition at $t_i + \Delta t$, the predictive control

algorithm optimizes J subject to the set of linear inequality constraints to find the new $\dot{u}(t_i + \Delta t)$. Therefore, the optimization is required to be completed within the allocated Δt time to avoid any computational delay. If the computational algorithm is sufficiently fast to produce $\dot{u}(t_i + \Delta t)$ on time when the real-time clock ticks at $t_i + \Delta t$, then there will not be a computational delay. However, if the computational speed is not matched with the fast sampling rate, then $\dot{u}(t_i + \Delta t)$ is produced with the time $\Delta t + \Delta c$, and Δc is the computational delay.

Strategy to Deal with Computational Delay

When computational delay occurs, the implementation clock time ticks at $t_i + \Delta t$ and the optimal coefficient vector η is not available. To distinguish the difference, let us call η_c the vector η computed using the information $\hat{x}(t_i + \Delta t)$ and $r(t_i + \Delta t)$, and η_p the vector η computed using the information $\hat{x}(t_i)$ and $r(t_i)$. A common strategy is to replace the optimal control derivative

$$\dot{u}(t_i + \Delta t) = L(0)^T \eta_c, \quad (7.32)$$

where η_c is still being computed, by the extended optimal control derivative from the previously computed η_p with the expression:

$$\dot{u}(t_i + \Delta t) = L(\Delta t)^T \eta_p. \quad (7.33)$$

This is not an optimal solution, but a sub-optimal solution. The difference is that $\dot{u}(t_i + \Delta t)$ used the estimated state variable $\hat{x}(t_i)$, instead of $\hat{x}(t_i + \Delta t)$. However, we argue that there are two scenarios related to this sub-optimal solution.

The first scenario is the case where there is no external set-point change or disturbance occurring at time t_i or during the time from t_i to $t_i + \Delta t$. Another important underlying assumption is that the model (A, B, C) is an accurate representation of the dynamic system. This statement actually means that the prediction generated by the model is sufficiently close to the actual output. If this happens, then

$$\hat{x}(t_i + \Delta t) = (A\hat{x}(t_i) + B\dot{u}(t_i)) \Delta t + \hat{x}(t_i); \quad r(t_i + \Delta t) = r(t_i). \quad (7.34)$$

The predictive control has taken this prediction into account in its design, thus the optimal control at time $t_i + \Delta t$ is identical to $L(\Delta t)^T \eta_p$ and there is no compromise in the solution. We have demonstrated similar cases in the examples presented in the discrete-time case (see Chapters 3 and 4).

The second scenario is the case where there are external set-point changes or disturbances occurring at time t_i or during the time from t_i to $t_i + \Delta t$. Or the predicted response from the model is different from the actual system response. Then, the information presented in (7.34) is no longer accurate. There will be a difference between the solutions obtained from $L(0)^T \eta_c$ and

from $L(\Delta t)^T \eta_p$. The changes in the external signals could not be contained in the information $L(\Delta t)^T \eta_p$. Thus, the computational delay appears as a time delay in the measurement. We emphasize that if computational delay is anticipated, then the constraints need to be imposed on the time intervals where the extended control trajectory will be used. For instance, if we prepare to extend the solution at Δt , then the constraints will be imposed at least at 0 and Δt so to ensure that they are satisfied when computational delay occurs and η_p is used to replace the unavailable optimal solution η_c .

7.5 Summary

This chapter has discussed continuous-time predictive control with constraints. Similar to the discrete-time counterpart, the constraints in the continuous-time MPC are also placed on the current and future values of the control signal, and the derivative of the control signal. In addition, if desired, constraints on the future state variable and plant output are introduced. The constrained control problem in continuous time is formulated as a minimization of a quadratic cost function subject to linear inequality constraints, which is a quadratic programming problem. Similar to the discrete-time case, the coefficients of the Laguerre functions are the decision variables. In the solution, Hildreth's quadratic programming procedure is used to identify the active constraints, and subsequently the optimal decision variable η is found. When the set of active constraints can be correctly guessed, the decision variable η is obtained using the closed-form solution (see Chapter 2). This is particularly useful in the continuous-time case, as a fast sampling rate is often used to take advantage of the continuous-time setting.

One of the key differences between continuous-time and discrete-time MPC systems is that the orthonormal basis functions are in the continuous-time domain. Therefore, when introducing constraints on the future trajectories, a set of fixed nodes are chosen as the discretized time intervals for the constraints to be enforced upon. If the nodes were selected to be too close to each other, then constraints could be redundant. Therefore, a sensible choice needs to be made from application to application. In any case, the constraints are always enforced at the current time, and perhaps, one or two nodes are sufficient for the future time.

When the sampling rate in the implementation is faster than the computational rate dictated by the quadratic programming procedure, a computational delay occurs. A natural way to deal with the computational delay is to extend the optimal control trajectory obtained from previous computation to the current sample time, through the Laguerre functions and the previous optimal Laguerre coefficients. Like any other predictor-based approach, this strategy will work well if the modelling error between the plant and the model is small. An application of this approach to an inverted pendulum was pre-

sented in Gawthrop and Wang (2006). The approach was termed intermittent model predictive control in Gawthrop and Wang (2007).

Problems

7.1. A DC motor with speed as output variable and voltage as input variable has a first-order transfer function model given by

$$G(s) = \frac{1}{s+1}$$

Design a continuous-time predictive controller with constraints for this DC motor. Assuming zero initial conditions, the output response is required to follow a positive and a negative step input change where the constraints are specified as

$$-0.5 \leq \dot{u}(t) \leq 0.5; -1 \leq u(t) \leq 1.$$

The other performance parameters are $Q = C^T C$, $R = 1$, $N = 2$, $p = 0.5$, and the prediction horizon $T_p = 3$. Sampling interval $\Delta t = 0.01$.

1. Impose the constraints on the first sample of the signal and solve the constrained control problem analytically.
2. Simulate the constrained MPC system with zero initial conditions, and a positive unit step reference change at $t = 0$ and a negative unit step change at $t = 5$.
3. Show that without constraints the control signal can be written as components of proportional and integral controls,

$$u(t) = k_1 \int (r(\tau) - y(\tau)) d\tau - k_2 y(t),$$

where k_1 and k_2 are the gains of the predictive control system. Present schematically the configuration of the predictive control system, which is identical to a traditional proportional integral (PI) control system (see Astrom and Hagglund, 1995, Johnson and Moradi, 2005).

7.2. Assume that a second-order system has the continuous-time transfer function

$$G(s) = \frac{b_0}{s^2 + a_1 s + a_0}.$$

Verify that the augmented state-space model using $x_1(t) = \ddot{y}(t)$, $x_2(t) = \dot{y}(t)$ and $x_3(t) = y(t)$ has the following form:

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} -a_1 & -a_0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} b_0 \\ 0 \\ 0 \end{bmatrix} \dot{u}(t) \\ y(t) &= [0 \ 0 \ 1]. \end{aligned} \tag{7.35}$$

1. Show that a continuous-time MPC based on this model structure has the control signal in the form:

$$u(t) = k_1 \int (r(\tau) - y(\tau)) d\tau - k_2 y(t) - k_3 \dot{y}(t),$$

which is equivalent to a proportional integral derivative (PID) controller.

2. Suppose that $a_1 = 1$, $a_0 = 0.1$ and $b_0 = 1$. Find the predictive controller parameters for $Q = C^T C$, $R = 0.1$, $p = 0.5$, $N = 2$, and $T_p = 20$.
3. Implement this predictive control system with constraints in a closed-form solution, assuming sampling interval $\Delta t = 0.01$, step reference input signal, zero initial conditions. The constraints are

$$-0.1 \leq \dot{u}(t) \leq 0.1; \quad -0.1 \leq u(t) \leq 0.1.$$

4. In the implementation of a PID controller (see Astrom and Hagglund, 1995), a derivative filter is often needed because of the presence of measurement noise. Similarly, you could use a derivative filter in the implementation here by using the relationship

$$k_3 s Y(s) \approx \frac{k_3 s}{\beta k_3 s + 1} Y(s),$$

where k_3 is the state feedback gain corresponding to $\dot{y}(t)$, and β is 0.1 or less. Use of an observer is another option in the presence of measurement noise. Positioning observer poles at $-2, -2.1$ and -2.2 , implement the predictive control system. Compare the implementations in the presence of measurement noise that is simulated using a sequence of zero-mean white noise with standard deviation of 0.1.

7.3. A distillation column is described by the transfer function model

$$G(s) = \begin{bmatrix} \frac{0.66e^{-2.6s}}{6.7s+1} & \frac{-0.51e^{-3s}}{8s+1} & \frac{-0.3e^{-s}}{10s+1} \\ \frac{1.5e^{-8s}}{4.1s+1} & \frac{-5e^{-2s}}{5.4s+1} & \frac{-1.3e^{-2s}}{9s+1} \\ \frac{-0.3e^{-11s}}{8s+1} & \frac{0.1e^{-10.8s}}{9s+1} & \frac{0.9e^{-2s}}{50s+1} \end{bmatrix}, \quad (7.36)$$

where typically the outputs are product concentration and product purity, and the inputs are the feed flow, temperature and vaporization rate. A case study of a distillation train was given in (Wang and Cluett, 2000) for the purpose of mathematical modelling.

1. Approximate the time delays in the transfer function model using the second-order Pade approximation where $e^{-ds} = \frac{(ds-4)^2}{(ds+4)^2}$, and find a minimal state-space realization (A_m, B_m, C_m) of the transfer function model using MATLAB functions ‘tf’ and ‘ss’ (see e.g., Tutorial 3.3).

2. Design a continuous-time MPC system that will reject a constant input disturbance and follow a constant setpoint change without steady-state errors. The performance specifications are $Q = C^T C$, $R = 0.2I$ where I is the 3×3 identity matrix, $T_p = 50$. $N_1 = 3$, $N_2 = 3$ and $N_3 = 3$. $p_1 = 1/6$, $p_2 = 1/5$, $p_3 = 1/40$, which are close to the dominant poles of the diagonal elements in $G(s)$. The Laguerre scaling parameters should not be equal to the poles in the model as the inverse of matrix $A + pI$ is required in the solution of convolution integral equations (see section 6.3.3).
3. An observer is required in the implementation, where the observer gain K_{ob} is found by using the MATLAB lqr function with the pair A^T, C^T and the weight matrices are $Q_{ob} = I$ and $R_{ob} = 0.1I$.
4. Simulate the nominal closed-loop performance without constraints, with zero initial conditions and a unit step reference input for y_2 at time 0 and a unit input disturbance added to $u_1(t)$ at time 60. The reference signals for y_1 and y_3 are zero. The sampling interval for implementation is 0.01.
5. It is relatively easy to build a SIMULINK simulation program when the plant has time delays. Writing the predictive control system as state feedback control $\dot{u}(t) = -K_{mpc}x(t)$, without constraints, simulate the predictive control system with the plant model that contains the time delays and compare the closed-loop responses with the nominal closed-loop responses.

7.4. Continue from Problem 7.3, where we will design and implement constrained MPC for the distillation column. The objective of the constraints is to maintain plant input and output within a desired operating region when performing set-point changes and rejecting disturbances originating from plant operations such as upstream feed flow changes. Assuming zero steady state for the system, introduce unit step change at $y_2(t)$ at time $t = 0$, then introduce unit step input disturbance at $u_1(t)$ at time $t = 30$. The operational constraints are specified as

$$\begin{aligned} -1.3 &\leq u_1 \leq 0, \quad -0.3 \leq u_2 \leq 0.1, \quad -0.4 \leq u_3 \leq 0.1 \\ -0.1 &\leq \dot{u}_1(t) \leq 0.05, \quad -0.1 \leq \dot{u}_2(t) \leq 0.1, \quad -0.05 \leq \dot{u}_3(t) \leq 0.05. \end{aligned}$$

Realize the constrained control using Hildreth's quadratic programming procedure. Impose the constraints on the first sample of the control signals in the first set of simulation experiments; then impose constraints on the first sample, $\tau_0 = 0$ and the second sample $\tau_1 = 1$.

7.5. Continue from Problem 7.4. The sampling interval for implementation of the constrained control is 0.01 in the simulation study. In the actual real-time implementation, the solution of the quadratic programming problem may require a longer computational time. Assume that the computational time on average is about 20 times the Δt used in the simulation. Revise the constrained predictive control scheme to cope with this computational delay using the technique introduced in Section 7.4. Compare the results obtained from using this predictive control to the results obtained in Problem 7.4.

Continuous-time MPC with Prescribed Degree of Stability

8.1 Introduction

This chapter will propose a set of continuous-time model predictive control algorithms that are numerically stable and have a prescribed degree of stability. Section 8.2 begins with an example of the control of an unstable system, demonstrating that when the prediction horizon increases, the original approaches to continuous-time MPC design described in Chapter 6 will lead to an ill-conditioned Hessian matrix. This problem is caused by the open-loop prediction using the unstable model in addition to the embedded integrator(s) in the system matrix A for integral action. In Section 8.3, we show a strategy to overcome this by using a stable matrix A for the design, which is achieved by using an exponential weight in the cost function. This essentially transforms the original state and derivative of the control variables into exponentially weighted variables for the optimization procedure. In Section 8.4, we move on to the next step that produces a model predictive control system with infinite prediction horizon with asymptotic stability. With a slight modification on the weight matrices, a prescribed degree of stability can be achieved in the design of model predictive control (see Section 8.5). The final section discusses how constraints are introduced in the design (see Section 8.6). The stability results in this chapter are all based on the assumption of a sufficiently large prediction horizon T_p used in the design.

8.2 Motivating Example

This section examines an example based on the design algorithms introduced in Chapter 6. It emphasizes that because the design model is unstable, the algorithms are numerically ill-conditioned for a large prediction horizon T_p . This problem is particularly severe for systems that contain unstable poles, as illustrated in the example below.

Example 8.1. A dynamic system is described by the state-space model given as

$$\begin{aligned}\dot{x}(t) &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{\alpha_1}{V} & \frac{\beta_1}{V} & 0 \\ 0 & 0 & -a & a \\ 0 & 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= [1 \ 0 \ 0 \ 0] x(t),\end{aligned}\quad (8.1)$$

where $\alpha_1 = 10.2$, $\beta_1 = 0.32$, $a = 72$ and $V = 60$. This system has four open-loop eigenvalues as

$$[0 \ 0 \ 0.17 \ -72].$$

Design a continuous-time model predictive control for this system using the algorithm presented in Chapter 6. The design parameters are $p = 0.8$, $N = 4$, and $R = 0.1$, however, the prediction horizon T_p should be used as a tuning parameter. The design objective is for reference following of a step input signal.

Solution. We change the prediction horizon T_p and observe what happens with respect to the closed-loop performance and numerical condition of the algorithm.

Case A. The case of a short prediction horizon $T_p = 10$ is examined. With this choice the closed-loop control system is unstable, which is indicated by the location of the closed-loop eigenvalues

$$[-71.99 \ -2.883 \ 0.0554 \pm j0.166 \ -0.088],$$

where the pair of complex poles are on the right half of the complex plane. The condition number of the Hessian matrix is 146.68, which is irrelevant because the predictive control system is unstable.

Case B. The prediction horizon T_p is selected to be 13, which is increased on the one used in Case A. The Hessian matrix is

$$\Omega = \begin{bmatrix} 85.9886 & -57.8910 & 37.0031 & -21.7936 \\ -57.8910 & 47.8025 & -35.1036 & 21.6875 \\ 37.0031 & -35.1036 & 31.2090 & -21.6632 \\ -21.7936 & 21.6875 & -21.6632 & 18.8841 \end{bmatrix}.$$

The closed-loop control system is stable, which is seen from the location of the closed-loop eigenvalues

$$[-71.99 \ -2.7355 \ -0.0334 \pm j0.1568 \ -0.1394].$$

The state feedback control gain obtained from the predictive control is

$$K_{mpc} = [18.6 \ 212.24 \ 0.0164 \ 3.11 \ 1.837].$$

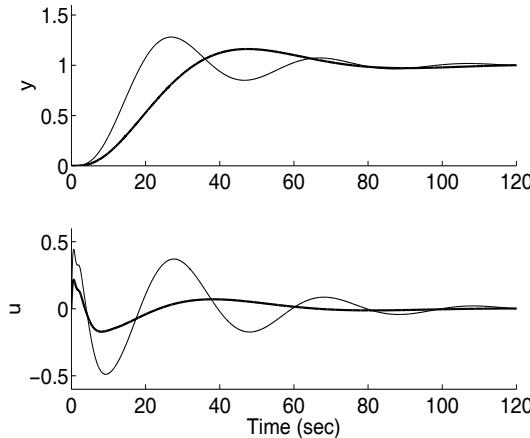


Fig. 8.1. Comparison of closed-loop responses for Case B (solid-line) and Case C (darker-solid-line)

For this choice of prediction horizon, the condition number for the Hessian matrix is 240.8.

Case C. The case of long prediction horizon T_p is examined. Let us choose $T_p = 50$. The Hessian matrix is

$$\Omega = 10^8 \times \begin{bmatrix} 1.4102 & -0.9097 & 0.5854 & -0.3754 \\ -0.9097 & 0.5869 & -0.3776 & 0.2422 \\ 0.5854 & -0.3776 & 0.2430 & -0.1558 \\ -0.3754 & 0.2422 & -0.1558 & 0.0999 \end{bmatrix}.$$

The closed-loop control system is stable with the location of the closed-loop eigenvalues as

$$[-71.99 - 2.6016 - 0.0412 \pm j0.0737 - 0.2616],$$

and the state feedback control gain obtained from the predictive control scheme is

$$K_{mpc} = [14.355 \ 273.33 \ 0.0208 \ 3.115 \ 0.909].$$

Figure 8.1 shows the comparison results of the closed-loop responses for Case B and Case C. It is seen that the closed-loop output response from Case C is less oscillatory than the one from Case B. Although the closed-loop response is satisfactory when $T_p = 50$, the condition number for the Hessian has increased from 240.8 to 3.11×10^8 . It is clear that the predictive control scheme is numerically ill-conditioned for Case C.

This example shows that the predictive control algorithm is sensitive to the choice of prediction horizon. If the prediction horizon is short, the closed-loop

control system could become unstable; however, if the prediction horizon is long, then the algorithm would become numerically ill-conditioned. Despite these sensitivities in the predictive control algorithms, they have still gained acceptance by the process industry. This is mainly due to their simplicity and easy-to-implement features.

For the rest of the chapter, the continuous-time predictive control algorithms presented in Chapter 6 will be modified to achieve the three objectives: (1) removing the numerical ill-condition problem from the design when the prediction horizon T_p is large; (2) deriving a design that will lead to asymptotic closed-loop stability for a large prediction horizon; (3) providing a solution that will have a prescribed degree of stability. Perhaps, above all, the key features of the model predictive control algorithms will be maintained to be simple and easy-to-implement.

8.3 CMPC Design Using Exponential Data Weighting

From the analysis, we can see that the model predictive control algorithm became numerically ill-conditioned when the prediction horizon T_p became large. The reason for this is that the system matrix used for prediction contains eigenvalues with positive real parts and an integrator, which leads to

$$\|e^{At}\| \rightarrow \infty,$$

as $t \rightarrow \infty$.

In this section, we explore the exponential data weighting strategy used in Anderson and Moore (1971) to produce a predictive control algorithm that is numerically well-conditioned. To begin, let us define a cost function to be optimized by the predictive control as

$$J = \int_0^{T_p} [e^{-2\alpha\tau} x(t_i + \tau | t_i)^T Q x(t_i + \tau | t_i) + e^{-2\alpha\tau} \dot{u}(\tau)^T R \dot{u}(\tau)] d\tau, \quad (8.2)$$

subject to

$$\dot{x}(t_i + \tau | t_i) = Ax(t_i + \tau | t_i) + Bu(\tau),$$

with initial condition $x(t_i)$. If the set-point signal is non-zero, as before, the last block variables in $x(t_i + \tau | t_i)$ correspond to the error signals between the output and set-point, which is translated to the difference in initial condition $x(t_i)$, while the rest of the formulations remain unchanged.

As before, Q is a symmetric nonnegative definite matrix, and R is a symmetric positive definite matrix. The constant α can be either positive or negative or equal to zero, depending on the application. The exponential weight used by Anderson and Moore was $e^{-2\alpha\tau}$ with α negative, which effectively produces an exponentially increasing weight. Their results were to produce an optimal regulator with prescribed degree of stability (see Section 8.5). Our interest here is to use a positive α that effectively produces an exponentially decreasing weight.

Minimization of Exponentially Weighted Cost

The results of $\alpha \geq 0$ are summarized in the theorem as follows.

Theorem 8.1. *For a given $\alpha \geq 0$, $T_p > 0$, $Q \geq 0$, and $R > 0$, minimization of the cost function*

$$J_1 = \int_0^{T_p} [e^{-2\alpha\tau} x(t_i + \tau | t_i)^T Q x(t_i + \tau | t_i) + e^{-2\alpha\tau} \dot{u}(\tau)^T R \dot{u}(\tau)] d\tau, \quad (8.3)$$

subject to

$$\dot{x}(t_i + \tau | t_i) = Ax(t_i + \tau | t_i) + B\dot{u}(\tau); \quad x(t_i | t_i) = x(t_i),$$

is equivalent to minimization of

$$J = \int_0^{T_p} [x_\alpha(t_i + \tau | t_i)^T Q x_\alpha(t_i + \tau | t_i) + \dot{u}_\alpha(\tau)^T R \dot{u}_\alpha(\tau)] d\tau, \quad (8.4)$$

subject to

$$\dot{x}_\alpha(t_i + \tau | t_i) = (A - \alpha I)x_\alpha(t_i + \tau | t_i) + B\dot{u}_\alpha(\tau); \quad x_\alpha(t_i | t_i) = x(t_i | t_i) = x(t_i),$$

where $x_\alpha(\cdot)$ and $\dot{u}_\alpha(\cdot)$ are the exponentially weighted variables of $x(\cdot)$ and $\dot{u}(\cdot)$

$$x_\alpha(t_i + \tau | t_i) = e^{-\alpha\tau} x(t_i + \tau | t_i); \quad \dot{u}_\alpha(\tau) = e^{-\alpha\tau} \dot{u}(\tau).$$

Proof. The cost function J_1 (8.3) equals the cost function J (8.4) with the transformed variables $x_\alpha(\cdot)$ and $\dot{u}_\alpha(\cdot)$. In addition

$$\dot{x}(t_i + \tau | t_i) = \frac{d e^{\alpha\tau} x_\alpha(t_i + \tau | t_i)}{d\tau}$$

$$= \alpha e^{\alpha\tau} x_\alpha(t_i + \tau | t_i) + e^{\alpha\tau} \dot{x}_\alpha(t_i + \tau | t_i) \quad (8.5)$$

$$= Ax(t_i + \tau | t_i) + B\dot{u}(\tau). \quad (8.6)$$

Therefore, by multiplying both (8.5) and (8.6) with $e^{-\alpha\tau}$, and re-arranging, we obtain the following equation

$$\dot{x}_\alpha(t_i + \tau | t_i) = (A - \alpha I)x_\alpha(t_i + \tau | t_i) + B\dot{u}_\alpha(\tau),$$

with the identical initial condition at $\tau = 0$,

$$x_\alpha(t_i | t_i) = x(t_i | t_i).$$

Use of the Results in CMPC Design

Theorem 8.1 shows that if we use a cost function that contains exponential decay weight, which is a time-varying weight, then the optimal solution is found by minimizing a cost function that has eliminated the time-varying weight. However, the state-space system matrix A is shifted by a $-\alpha I$ matrix. If α is positive, then all eigenvalues of original A matrix are shifted by the scalar $-\alpha$ to yield the eigenvalues of the matrix $A - \alpha I$, which effectively changes the real part of all eigenvalues. In this transformed formulation, the α value is selected such that the eigenvalues of $A - \alpha I$ lie strictly on the left-half of the complex plane, *i.e.*, $\text{real}(\lambda_i(A - \alpha I)) < -\epsilon$ for $\epsilon > 0$ and all i . Thus, the continuous-time model used for the design then becomes a stable model, instead of the unstable model in the original formulation. As a result, the numerical ill-conditioning problem is overcome.

Denote $A_\alpha = A - \alpha I$. At time t_i , $x_\alpha(t_i | t_i) = x(t_i)$ and the transformed derivative of the control signal is

$$\dot{u}_\alpha(\tau) = [L_1(\tau)^T \ L_2(\tau)^T \ \dots \ L_m(\tau)^T] \eta.$$

The predicted, transformed state variable $x_\alpha(\tau | t_i)$ at time τ is

$$\begin{aligned} x_\alpha(t_i + \tau | t_i) &= e^{A_\alpha \tau} x(t_i) \\ &+ \int_0^\tau e^{A_\alpha(\tau-\gamma)} [B_1 L_1(\gamma)^T \ B_2 L_2(\gamma)^T \ \dots \ B_m L_m(\gamma)^T] d\gamma \eta, \end{aligned} \quad (8.7)$$

where we assume that the number of inputs is m and Laguerre functions are used in the parameterization of the derivative of the control signal. Namely, we optimize the transformed control variable $\dot{u}_\alpha(\tau)$, instead of the original variable $\dot{u}(\tau)$. Introducing

$$\phi_i(\tau)^T = \int_0^\tau e^{A_\alpha(\tau-\gamma)} B_i L_i(\gamma)^T d\gamma, \quad (8.8)$$

and

$$\phi(\tau)^T = [\phi_1(\tau)^T \ \phi_2(\tau)^T \ \dots \ \phi_m(\tau)^T].$$

Equation (8.7) is simplified into

$$x_\alpha(t_i + \tau | t_i) = e^{A_\alpha \tau} x(t_i) + \phi(\tau)^T \eta. \quad (8.9)$$

Substituting (8.9) into the cost function (8.4), we obtain

$$J = \int_0^{T_p} (\eta^T \phi(\tau) Q \phi(\tau)^T \eta + 2\eta^T \phi(\tau) Q e^{A_\alpha \tau} x(t_i)) d\tau + \eta^T R_L \eta + \text{constant}, \quad (8.10)$$

where R_L is a block diagonal matrix with the k th block being R_k , and $R_k = r_{wk} I_{N_k \times N_k}$ (where $I_{N_k \times N_k}$ is a unit matrix with dimension N_k). Here,

for simplicity of the solution, we have assumed that the weight matrix R is a diagonal matrix.

Defining the data matrices

$$\Omega = \int_0^{T_p} \phi(\tau) Q \phi(\tau)^T d\tau + R_L \quad (8.11)$$

$$\Psi = \int_0^{T_p} \phi(\tau) Q e^{A_\alpha \tau} d\tau, \quad (8.12)$$

the quadratic cost function (8.10) becomes

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(t_i) + \text{constant}. \quad (8.13)$$

The optimal solution that minimizes the above quadratic cost function is

$$\eta = -\Omega^{-1} \Psi x(t_i). \quad (8.14)$$

Upon obtaining η , the exponentially weighted derivative of the control signal $\dot{u}_\alpha(\tau)$ is constructed through

$$\dot{u}_\alpha(\tau) = [L_1(\tau)^T \ L_2(\tau)^T \ \dots \ L_m(\tau)^T] \eta. \quad (8.15)$$

From the receding horizon control, the optimal solution for the actual $\dot{u}(0)$ is

$$\dot{u}(0) = \dot{u}_\alpha(0) = [L_1(0)^T \ L_2(0)^T \ \dots \ L_m(0)^T] \eta. \quad (8.16)$$

Because the optimization is performed on the transformed variables $x_\alpha(\cdot)$ and $\dot{u}_\alpha(\cdot)$, when constraints are introduced, all the original constraints are required to be transformed from the variables $x(\cdot)$ and $\dot{u}(\cdot)$ to $x_\alpha(\cdot)$ and $\dot{u}_\alpha(\cdot)$. Constrained control will be discussed further in the later sections of the chapter.

8.4 CMPC with Asymptotic Stability

This section establishes equivalent results with LQR when exponential weighting is used. The results are investigated through two different cost functions, and we then establish that the optimal control results are identical. The results are summarized in the theorem as follows.

Case A

Suppose that the optimal control $\dot{u}_1(\tau)$ is obtained by minimizing cost function J_1 with $Q \geq 0$, and $R > 0$

$$J_1 = \int_0^\infty [x(t_i + \tau | t_i)^T Q x(t_i + \tau | t_i) + \dot{u}(\tau)^T R \dot{u}(\tau)] d\tau, \quad (8.17)$$

subject to

$$\dot{x}(t_i + \tau | t_i) = Ax(t_i + \tau | t_i) + Bu(\tau),$$

where the initial state is $x(t_i)$. The optimal solution of the derivative of the control $\dot{u}(\tau)$ is obtained through the state feedback control law

$$\dot{u}_1(\tau) = -R^{-1}BPx(t_i + \tau | t_i), \quad (8.18)$$

and P is the solution of the Riccati equation

$$PA + A^T P - PBR^{-1}B^T P + Q = 0. \quad (8.19)$$

Case B

Choosing $Q_\alpha = Q + 2\alpha P$, $\alpha > 0$, R unchanged, the optimal control $\dot{u}_2(\tau)$ is obtained by minimizing

$$J_2 = \int_0^\infty e^{-2\alpha\tau} (x(t_i + \tau | t_i)^T Q_\alpha x(t_i + \tau | t_i) + \dot{u}(\tau)^T R \dot{u}(\tau)) d\tau, \quad (8.20)$$

subject to

$$\dot{x}(t_i + \tau | t_i) = Ax(t_i + \tau | t_i) + Bu(\tau),$$

with the initial condition $x(t_i)$.

Theorem 8.2. *The optimal control solutions stated in Case A and Case B have the following relation:*

$$\dot{u}_2(\tau) = \dot{u}_1(\tau); \min(J_2) = \min(J_1).$$

Proof. The optimal solution for Case A is found through the algebraic Riccati equation (Kailath, 1980, Bay, 1999)

$$PA + A^T P - PBR^{-1}B^T P + Q = 0, \quad (8.21)$$

with $\dot{u}_1(\tau) = -R^{-1}BPx(t_i + \tau | t_i)$ and $\min(J_1) = x(t_i)^T Px(t_i)$.

By adding and subtracting the term $2\alpha P$, (8.21) becomes

$$PA + A^T P - PBR^{-1}B^T P + Q + 2\alpha P - 2\alpha P = 0, \quad (8.22)$$

which is

$$P(A - \alpha I) + (A - \alpha I)^T P - PBR^{-1}B^T P + Q + 2\alpha P = 0. \quad (8.23)$$

With $Q_\alpha = Q + 2\alpha P$, the Riccati equation (8.23) becomes identical to

$$P(A - \alpha I) + (A - \alpha I)^T P - PBR^{-1}B^T P + Q_\alpha = 0. \quad (8.24)$$

Relating these back to the exponential data weighting results in Theorem 8.1, (8.24) is the Riccati equation for the optimization of Case B. Since (8.24) is identical to (8.21), therefore, the Riccati solution P from (8.23) remains unchanged, and hence

$$\dot{u}_2(\tau) = \dot{u}_1(\tau); \min(J_1) = \min(J_2).$$

The original Case A is not solvable in the context of predictive control for a sufficiently large prediction horizon T_p , when the design model contains eigenvalues on the imaginary axis or on the right half of the complex plane, and the prediction becomes numerically ill-conditioned. In contrast, the Case B is solvable in the context of predictive control because of the choice of the exponential weight $\alpha > 0$ that will lead to the system matrix $A - \alpha I$ becoming stable.

The following list summarizes the relationship between the design parameters and variables in LQR and the continuous-time MPC with exponential data weighting for sufficiently large N and T_p .

Model (LQR)	$\dot{x}(t_i + \tau t_i) = Ax(t_i + \tau t_i) + B\dot{u}(\tau)$
Model (CMPC)	$\dot{x}_\alpha(t_i + \tau t_i) = (A - \alpha I)x_\alpha(t_i + \tau t_i) + B\dot{u}_\alpha(\tau)$
Weight matrices (LQR)	Q, R
Weight matrices (CMPC)	$Q_\alpha = Q + 2\alpha P$, R unchanged
Cost (LQR)	$J = \int_0^\infty (x(\cdot)^T Q x(\cdot) + \dot{u}(\tau)^T R \dot{u}(\tau)) d\tau$
Cost (CMPC)	$J = \int_0^{T_p} (x_\alpha(\cdot)^T Q_\alpha x_\alpha(\cdot) + \dot{u}_\alpha(\tau)^T R \dot{u}_\alpha(\tau)) d\tau$
Optimal control (LQR)	$\dot{u}(\tau) = -R^{-1}B^T P x(t_i + \tau t_i)$
Optimal control (CMPC)	$\dot{u}(\tau) = -L(\tau)^T \Omega^{-1} \Psi x(t_i)$
$0 \leq \tau \leq T_p$	$\dot{u}_\alpha(\tau) = \dot{u}(\tau) e^{-\alpha\tau}$
$0 \leq \tau \leq T_p$	$x_\alpha(t_i + \tau t_i) = x(t_i + \tau t_i) e^{-\alpha\tau}$
Feedback gain	$K_{mpc} = K_{lqr}$
Closed-loop Eigenvalues.	$\lambda_i(A - BK_{mpc}) = \lambda_i(A - BK_{lqr})$, for all i .

Tutorial 8.1. We consider the same system as in Example 8.1, where the augmented dynamic system with an integrator is described by the state-space model given as below

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{\alpha_1}{V} & \frac{\beta_1}{V} & 0 & 0 \\ 0 & 0 & -a & a & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \dot{u}(t) \\ y(t) &= [0 \ 0 \ 0 \ 0 \ 1] x(t), \end{aligned} \quad (8.25)$$

where $\alpha_1 = 10.2$, $\beta_1 = 0.32$, $a = 72$ and $V = 60$. The parameters in the Laguerre functions are selected as $p = 0.8$ and $N = 4$.

- Choosing $Q = C^T C$ and $R = 0.1$, design the LQR control system with the weight matrices Q and R , and find the Riccati equation solution P , the feedback gain matrix K and the closed-loop eigenvalues.

2. Using exponential data weighting in the cost function of predictive control with $\alpha = 0.18$ in the modified Q_α , compute the data matrices Ω and Ψ and verify the convergence of Ω and Ψ with respect to a large prediction horizon T_p .
3. Compare the exponentially weighted predictive control system with the LQR system.

Step by Step

1. Create a program called *exptut.m*.
2. We will first set-up the state-space model and the augmented state-space model. Enter the following the program into the file:

```
alpha1=10.2;
beta1=0.32;
a=72;
v=60;
Ap=[0 1 0 0; 0 alpha1/v beta1/v
0; 0 0 -a a; 0 0 0 0] ;
Bp=[0;0;0;1];
Cp=[1 0 0 0];
Dp=0;
[m1,n1]=size(Cp);
[n1,n_in]=size(Bp);
A=zeros(n1+m1,n1+m1);
A(1:n1,1:n1)=Ap;
A(n1+1:n1+m1,1:n1)=Cp;
B=zeros(n1+m1,n_in);
B(1:n1,:)=Bp;
C=zeros(m1,n1+m1);
C(:,n1+1:n1+m1)=eye(m1,m1);
```

3. Compute the LQR solution using the MATLAB ‘lqr’ function. K is the feedback gain, P is the Riccati equation solution and E is the set of closed-loop eigenvalues. Continue entering the following program into the file:

```
Q=C'*C;
R=0.1*eye(m1,m1);
[K,P,E]=lqr(A,B,Q,R);
```

4. Compute Q_α and specify the design parameters for the continuous-time predictive control system. We also modify the unstable system matrix A to stable A_α . A large N is used in this design to demonstrate that the results converge to the LQR system. You can choose a smaller N and discover the difference is small.

```
alpha=0.18;
Q_alpha=Q+2*alpha*P;
A_alpha=A-alpha*eye(n1+m1,n1+m1);
```

```
p=0.6;
N=10;
Tp=35;
[Omega,Psi]=cmpc(A_alpha,B,p,N,Tp,Q_alpha,R);
```

5. With Ω and Ψ matrices, the cost function for the on-line optimization is determined. With receding horizon control, the feedback control gain matrix K_{mpc} is calculated. Continue entering the following program into the file:

```
[A1,L0]=lagc(p,N);
K_mpc=L0'*(Omega\Psi);
A_cl=A-B*K_mpc;
E_mpc=eig(A_cl);
```

6. We need to verify the relationship between $u_\alpha(\tau)$ and $u(\tau)$; and the relationship between $x_\alpha(t_i + \tau | t_i)$ and $x(t_i + \tau | t_i)$. We solve for the Laguerre parameter vector η using an initial state variable, and construct the whole control trajectory $\dot{u}_\alpha(\cdot)$ using Laguerre functions. The trajectory of $x_\alpha(\cdot)$ is calculated by solving the differential equation. Continue entering the following program into the file:

```
N_sim=22000;
h=0.001;
X0=[0.1;0.2;0.3;0.4;0.5];
eta=-Omega\Psi*X0;
x=X0;
t=0:h:(N_sim-1)*h;
for kk=1:N_sim
    u_dot(kk)=(expm(A1*t(kk))*L0)'*eta;
    xs(:,kk)=x;
    x=x+(A_alpha*x+B*u_dot(kk))*h;
end
```

7. To compare the results, we also compute the LQR control trajectory and the trajectory of $x(\cdot)$. Continue entering the following program into the file:

```
A_lqr=A-B*K;
A_lqr_alpha=A_lqr-alpha*eye(n1+m1,n1+m1);
for kk=1:N_sim
    x_lqr(:,kk)=expm(A_lqr*t(kk))*X0;
    u_dot_lqr(kk)=-K*x_lqr(:,kk);
    x_lqr_alpha(:,kk)=expm(A_lqr_alpha*t(kk))*X0;
end
```

8. Run this program, then you will have the numerical results for the exponentially weighted continuous-time MPC system.

9. Increasing prediction horizon T_p , you will notice this parameter does not affect the control results after a certain large number.

Figure 8.2 shows the comparison results within one optimization window, between the variables in the original LQR systems and the transformed predictive control system using exponential weighting. For simplicity, instead of examining all components in x , we examine the last component in x , which is the output y . By visual inspection, we can see that the exponentially weighted variables decay faster. To show that indeed there is the factor of $e^{-\alpha\tau}$ difference between x_α and x , and u_α and u , we calculate the errors

$$e_y = \int_0^{22} (y_\alpha(\tau) - y(\tau)e^{-\alpha\tau})^2 d\tau = 5.5529 \times 10^{-4}, \quad (8.26)$$

$$e_u = \int_0^{22} (\dot{u}_\alpha(\tau) - \dot{u}(\tau)e^{-\alpha\tau})^2 d\tau = 6.1761 \times 10^{-6}. \quad (8.27)$$

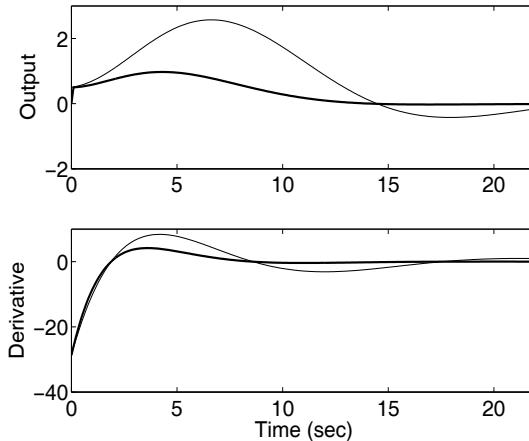


Fig. 8.2. Top figure: comparison between $y(\cdot)$ (solid-line) and $y_\alpha(\cdot)$ (darker-solid-line); bottom figure: comparison between $\dot{u}(\cdot)$ (solid-line) and $\dot{u}_\alpha(\cdot)$ (darker-solid-line)

Furthermore, the gain and closed-loop eigenvalues of LQR and continuous-time MPC systems are compared as below:

K_{mpc}	$[23.2346 \ 121.8435 \ 0.0090 \ 1.1371 \ 3.1639]$
K_{lqr}	$[23.2281 \ 121.8430 \ 0.0090 \ 1.1372 \ 3.1623]$
$eigenvalues(mpc)$	$-72.0000 \ -0.1411 \pm j0.3241 \ -0.3424 \pm j0.1334$
$eigenvalues(lqr)$	$-72.0000 \ -0.1411 \pm j0.3240 \ -0.3425 \pm j0.1334$

8.5 Continuous-time MPC with Prescribed Degree of Stability

The term ‘prescribed degree of stability of β ’ means that the closed-loop eigenvalues of the predictive control system reside to the left of the line $s = -\beta$ on the complex plane. This is very practical in the design of continuous-time predictive control systems. For instance, the value of β becomes part of the closed-loop performance specification. For a multi-input and multi-output system, tuning the predictive control system can be very time consuming. By specifying a degree of stability, the tuning process for a complex system can be simplified.

8.5.1 The Original Anderson and Moore’s Results

We resort to the wealth of literature on the linear quadratic regulator (LQR). In Anderson and Moore (1971), the cost function with ($\beta > 0$) is

$$J_1 = \int_0^{\infty} e^{2\beta t} [x(t)^T Q x(t) + \dot{u}(t)^T R \dot{u}(t)] dt, \quad (8.28)$$

and it is minimized subject to

$$\dot{x}(t) = Ax(t) + Bu(t). \quad (8.29)$$

The minimization of the cost function (8.28) produces a closed-loop system with a prescribed degree of stability determined by the value of β . Note that the choice of the weight exponent, β , has an opposite sign to what we proposed earlier, and let us call this an exponentially increasing weight.

To proceed further, denote

$$x_{\beta}(t) = e^{\beta t} x(t); \quad u_{\beta}(t) = e^{\beta t} u(t).$$

Then, the problem of minimizing (8.28) is equivalent to the minimization of the cost function:

$$J_2 = \int_0^{\infty} [x_{\beta}(t)^T Q x_{\beta}(t) + \dot{u}_{\beta}(t)^T R \dot{u}_{\beta}(t)] dt, \quad (8.30)$$

subject to

$$\dot{x}_{\beta}(t) = (A + \beta I)x_{\beta}(t) + B\dot{u}_{\beta}(t). \quad (8.31)$$

The optimal control is obtained through the solution of the algebraic Riccati equation to solve for the transformed system $(A + \beta I, B)$

$$P(A + \beta I) + (A + \beta I)^T P - PBR^{-1}B^T P + Q = 0, \quad (8.32)$$

$$\dot{u}_{\beta}(t) = -R^{-1}BPx_{\beta}(t). \quad (8.33)$$

However, the original control signal is

$$\dot{u}(t) = \dot{u}_\beta(t)e^{-\beta t} = -R^{-1}BPx(t). \quad (8.34)$$

Therefore, the feedback controller gain has the identical formula, except that the Riccati matrix P is solved using (8.32), where $(A + \beta I)$ is used to replace the original system matrix A .

The following points are given to establish that the closed-loop system has a prescribed degree of stability β .

1. If the pair (A, D) is observable where $Q = D^T D$, then $(A + \beta I, D)$ is observable; if the pair (A, B) is controllable, then $(A + \beta I, B)$ is controllable. The solution of the Riccati equation (8.32) leads to asymptotic stability of the closed-loop system for the pair $(A + \beta I, B)$. Namely

$$\|x_\beta(t)\| \rightarrow 0$$

as $t \rightarrow \infty$.

2. Note that

$$x(t) = e^{-\beta t}x_\beta(t).$$

This means that $x(t)$ decays at least as fast as the rate of $e^{-\beta t}$.

3. This establishes that the exponentially weighted cost function produces a closed-loop system with a prescribed degree of stability β .
4. The asymptotic stability of the closed-loop system for the pair $(A + \beta I, B)$ ensures that the closed-loop eigenvalues, for all k

$$\text{real}\{\lambda_k(A + \beta I - BK)\} < 0,$$

where $K = R^{-1}BP$. This means that the closed-loop eigenvalues for the pair (A, B) must be at least, for all k

$$\text{real}\{\lambda_k(A - BK)\} < -\beta.$$

5. This establishes that the closed-loop eigenvalues are on the left of the $s = -\beta$ line in the complex plane.

8.5.2 CMPC with a Prescribed Degree of Stability

Although the exponentially increasing weight proposed by Anderson and Moore produces a closed-loop system with a prescribed degree of stability, their solution was obtained through the Riccati equation (see (8.32)). If their approach was used in predictive control design, then numerical problems would arise. This is because the system matrix $A + \beta I$ (design model) has eigenvalues shifted further towards the right-half of the complex plane by a distance of β , and the prediction that uses this model will exponentially grow at least at a rate of β . This approach to obtain a prescribed degree of stability is re-developed in the context of predictive control design. The results are summarized as below.

Case A

Suppose that the optimal control $\dot{u}_1(\tau)$ is obtained by minimizing with $Q \geq 0$, $R > 0$, $\beta > 0$,

$$J_1 = \int_0^\infty e^{2\beta\tau} [x(t_i + \tau | t_i)^T Q x(t_i + \tau | t_i) + \dot{u}(\tau)^T R \dot{u}(\tau)] d\tau, \quad (8.35)$$

subject to

$$\dot{x}(t_i + \tau | t_i) = Ax(t_i + \tau | t_i) + B\dot{u}(\tau); \quad x(t_i | t_i) = x(t_i),$$

where A may contain eigenvalues that are either on the jw axis or on the right-half of the complex plane. The optimal solution of the derivative of the control $\dot{u}(\tau)$ is obtained through the state feedback law

$$\dot{u}_1(\tau) = -R^{-1}B^T P x(t_i + \tau | t_i), \quad (8.36)$$

and P is the solution of the Riccati equation

$$P(A + \beta I) + (A + \beta I)^T P - PBR^{-1}B^T P + Q = 0. \quad (8.37)$$

Case B

Choosing $\alpha > 0$, R unchanged, and

$$Q_\alpha = Q + 2(\alpha + \beta)P,$$

the optimal control $\dot{u}_2(\tau)$ is obtained by minimizing

$$J_2 = \int_0^\infty e^{-2\alpha\tau} [x(t_i + \tau | t_i)^T Q_\alpha x(t_i + \tau | t_i) + \dot{u}(\tau)^T R \dot{u}(\tau)] d\tau, \quad (8.38)$$

subject to

$$\dot{x}(t_i + \tau | t_i) = Ax(t_i + \tau | t_i) + B\dot{u}(\tau); \quad x(t_i | t_i) = x(t_i).$$

Theorem 8.3. *The optimal solutions given in Case A and Case B satisfy the following relation:*

$$\dot{u}_2(\tau) = \dot{u}_1(\tau); \quad \min(J_2) = \min(J_1).$$

Proof. The proof follows a similar procedure to that in the proof of Theorem 8.2.

From the Anderson and Moore's results, the optimal solution for Case A is found through the algebraic Riccati equation

$$P(A + \beta I) + (A + \beta I)^T P - PBR^{-1}B^T P + Q = 0, \quad (8.39)$$

with $\dot{u}_1(\tau) = -R^{-1}BPx(t_i + \tau | t_i)$ and $\min(J_1) = x(t_i)^T Px(t_i)$. By adding and subtracting the term $2\alpha P$, (8.39) becomes

$$P(A + \beta I) + (A + \beta I)^T P - PBR^{-1}B^T P + Q + 2\alpha P - 2\alpha P = 0, \quad (8.40)$$

which is

$$P(A - \alpha I) + (A - \alpha I)^T P - PBR^{-1}B^T P + Q + 2\alpha P + 2\beta P = 0. \quad (8.41)$$

With $Q_\alpha = Q + 2(\alpha + \beta)P$, the Riccati equation (8.41) becomes identical to

$$P(A - \alpha I) + (A - \alpha I)^T P - PBR^{-1}B^T P + Q_\alpha = 0. \quad (8.42)$$

Comparing the Riccati equation (8.42) to the exponential data weighting results in Theorem 8.1, (8.42) is the Riccati equation for the optimization Case B. Since (8.42) is identical to (8.39), therefore, the Riccati solution P from (8.42) remains unchanged, and hence

$$\dot{u}_2(\tau) = \dot{u}_1(\tau); \min(J_1) = \min(J_2).$$

8.5.3 Tuning Parameters and Design Procedure

Selection of the Exponential Weighting Factor

From a given augmented state-space model (A, B) , the eigenvalues of A are determined. If the plant is stable, then the unstable eigenvalues of A come from the integrators that have been embedded in the model. In this case, any $\alpha > 0$ will serve the purpose of exponential data weighting. However, if the plant is unstable with all its eigenvalues lying on the left of the ϵ line of the complex plane where $\epsilon > 0$, the parameter α is required to be at least greater than ϵ . In summary, the idea behind the selection of α is to make sure that the design model with $(A - \alpha I)$ is stable with all eigenvalues on the left-half of the complex plane.

Selection of Prediction Horizon

Once the exponential weight factor α is selected, the eigenvalues of the matrix $A - \alpha I$ are fixed. Since this matrix is stable with an appropriate choice of α , the prediction of the state variables is numerically sound. Thus, the prediction horizon T_p is selected sufficiently large to capture the transformed state variable response. In general, if the eigenvalues of $A - \alpha I$ were further away from the imaginary axis on the complex plane, then a smaller T_p would be required. However, some attention needs to be paid to the computation of Ω and Ψ matrices when discretization is used to recursively evaluate the integrals given in Section 6.3.4. In general, the discretization interval (h) for the computation should be smaller if the exponential weight factor α is used.

Choice of Weight Matrices in the Cost Functions

From the model formulation, the Q matrix is usually selected as $Q = C^T C$, which corresponds to minimization of integral squared output errors. This choice has been found to produce satisfactory closed-loop performance for set-point tracking of a reference signal. Weight matrix R is selected as a diagonal matrix, with each element weighting the corresponding control signal. For instance, if the influence of a particular control is to be reduced, then the corresponding diagonal element will be increased to reflect this intention.

Selection of Degree of Stability β

The closed-loop performance of a predictive control system so far is determined by the choice of Q and R matrices. The tuning could be very time consuming as it often requires finding the off-diagonal elements in Q and R to achieve satisfactory performance. This is often carried out in a trial-and-error manner. Now, with the additional parameter β that dictates the degree of stability, the closed-loop eigenvalues of the predictive control system are effectively positioned to some desired regions on the complex plane. This parameter is very useful in the closed-loop performance specification. For instance, β is related to the minimal decay rate of the closed-loop system. So we can use this parameter to specify the closed-loop response speed.

The Parameters in Laguerre Functions

When N increases, the predictive control trajectory converges to the underlying optimal control trajectory of the linear quadratic regulator. However, with a small N , the pole location p will affect the closed-loop response. The pair of parameters (p, N) can be used as a pair of fine tuning parameters for the closed-loop performance. Detailed discussion on the Laguerre parameters for the discrete-time counterpart is given in Chapter 4.

The Q_α Matrix

With the choice of β , which is the degree of stability, the Riccati equation is solved for the P matrix:

$$P(A + \beta I) + (A + \beta I)^T P - PBR^{-1}B^T P + Q = 0. \quad (8.43)$$

MATLAB script can be used for this solution:

```
[K,P,E] = lqr(A+beta*eye(n,n), B, Q, R);
```

Matrix Q_α is determined, with the values of α , β and P , using

$$Q_\alpha = Q + 2(\alpha + \beta)P.$$

The Modified Design Model

The augmented state-space model (A, B) is modified for use in the design. The matrix B is unchanged, however, the matrix A is modified to become

$$A - \alpha I$$

With this set of performance parameters (Q_α, R) and the design model $(A - \alpha I, B)$, the predictive control problem is converted back to the original problem stated in Chapter 6, thus the cost function for the predictive control system is expressed as a function of η

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(t_i) + \text{constant.} \quad (8.44)$$

8.6 Constrained Control with Exponential Data Weighting

The design of continuous-time predictive control using exponential data weighting is based on the transformed variables $x_\alpha(\cdot)$ and $\dot{u}_\alpha(\cdot)$. By using the transformed variables, a large prediction horizon is used to approximate the infinity horizon case so as to guarantee asymptotic stability or to achieve a prescribed degree of stability, and the numerical ill-conditioning problem is overcome. In the constraints handling, the design specification of the constraints is given for the original variables $x(\cdot)$ and $\dot{u}(\cdot)$, and these are required to be mapped into constraints with respect to the transformed variables $x_\alpha(\cdot)$ and $\dot{u}_\alpha(\cdot)$.

Constraints at $\tau = 0$

Since within one optimization window, at $\tau = 0$ the transformed variables $x_\alpha(t_i | t_i)$ and $\dot{u}_\alpha(0)$ are identical to the original variables $x(t_i | t_i)$ and $\dot{u}(0)$, there is no change for the constraints at the time $\tau = 0$.

Constraints at $\tau > 0$

As we know, the relationship between the transformed variables and the original variables is given as

$$\dot{u}_\alpha(\tau) = \dot{u}(\tau)e^{-\alpha\tau}; \quad x_\alpha(t_i + \tau | t_i) = x(t_i + \tau | t_i)e^{-\alpha\tau}.$$

Thus, supposing that the upper and lower limits of $\dot{u}(\tau)$ are specified as

$$du^{min} \leq \dot{u}(\tau) \leq du^{max},$$

with respect the transformed variable $\dot{u}_\alpha(t)$ within one optimization window, the constraints are mapped into the relation:

$$e^{-\alpha\tau}du^{min} \leq \dot{u}_\alpha(\tau) \leq du^{max}e^{-\alpha\tau}, \quad (8.45)$$

which is expressed, in terms of the decision variable η , as

$$e^{-\alpha\tau}du^{min} \leq L(\tau)\eta \leq du^{max}e^{-\alpha\tau}. \quad (8.46)$$

Similarly, the original constraints on the state variables, x^{min} and x^{max} , are transformed into

$$e^{-\alpha\tau}x^{min} \leq \overbrace{e^{A_{\alpha\tau}}x(t_i) + \phi(\tau)^T\eta}^{x_\alpha(t_i+\tau|t_i)} \leq x^{max}e^{-\alpha\tau}. \quad (8.47)$$

Since the transformed variables exponentially decay in a faster rate, the original constant bounds become exponentially decaying with respect to \dot{u}_α and x_α to form tighter bounds.

What we will do next is to transform the bounds on the original control signal to the bounds on the transformed variables. Note that the control signal with assumed zero initial condition is expressed as

$$u(\tau) = \int_0^\tau \dot{u}(\gamma)d\gamma. \quad (8.48)$$

By substituting $\dot{u}(\gamma) = \dot{u}_\alpha(\gamma)e^{\alpha\gamma} = L(\gamma)^T e^{\alpha\gamma}\eta$ into (8.48), we obtain

$$u(\tau) = \int_0^\tau L(0)^T e^{(A_p+\alpha I)^T\gamma} \eta d\gamma \quad (8.49)$$

$$= L(0)^T \left(e^{(A_p+\alpha I)^T\tau} - I \right) (A_p + \alpha I)^{-T} \eta. \quad (8.50)$$

Adding the first sample of the control signal, the bounds are expressed in terms of the decision variable η as

$$u^{min} \leq u(t_i - \Delta t) + L(0)^T \eta \Delta t + L(0)^T \left(e^{(A_p+\alpha I)^T\tau} - I \right) (A_p + \alpha I)^{-T} \eta \leq u^{max}. \quad (8.51)$$

Upon setting up the constraints with respect to the transformed variables, the remaining procedures to the solution of the constrained control problem are identical to those without exponential data weighting, as discussed in Chapter 7, namely, the inequality constraints are used in the optimization of η by minimizing the cost function:

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(t_i).$$

Example 8.2. Consider a two-input and two-output system described by the transfer function:

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{12.8(-s+4)^2}{(40s+1)(s+4)^2} & \frac{-10.9(-3s+4)^2}{(21.0s+1)(3s+4)^2} \\ \frac{12.8(-7s+4)^2}{(10.9s+1)(7s+4)^2} & \frac{-19.4(-3s+4)^2}{(20s+1)(3s+4)^2} \end{bmatrix} \begin{bmatrix} u_1(s) \\ u_2(s) \end{bmatrix}. \quad (8.52)$$

This system has complex unstable zeros and strong couplings as shown by the off diagonal elements of the transfer function. Choose the weight matrices $Q = C^T C$, $R = I$, and the prediction horizon $T_p = 50$, $N_1 = N_2 = 6$, and $p_1 = p_2 = 1$. An observer is needed in the implementation of the predictive control system. MATLAB function lqr is used with $Q_{ob} = I$ and $R = 0.2I$. With zero initial conditions on the state variables, for a unit set-point change, the operational constraints on the control signals are specified as

$$0 \leq u_1(t), u_2(t) \leq 0.3; -0.2 \leq \dot{u}_1(t), \dot{u}_2(t) \leq 0.4.$$

Design and simulate a continuous-time predictive control system with constraints using exponential data weighting, where $\alpha = 0.18$, and compare the results with the case when $\alpha = 0$. The sampling interval is selected as $\Delta t = 0.009$ sec.

Solution. The condition number of the Hessian matrix with exponential data weighting ($\alpha = 0.18$) is $\kappa(\Omega) = 720$. In contrast, the condition number without exponential data weighting ($\alpha = 0$) is $\kappa(\Omega) = 1.474 \times 10^5$, which clearly indicates that the Hessian matrix is ill-conditioned. Table 8.3 shows the comparison between the elements of the first row in the state feedback gain matrix with three different approaches. It is seen that with exponential weighting ($\alpha = 0.18$), the elements of the predictive feedback control gain K_{mpc} are very close to the elements of feedback gain from LQR design. However, without exponential data weighting, there are large differences between the elements of the predictive controller gain and those from LQR design. We also confirm the large differences between the closed-loop responses from using exponential data weighting and not using exponential weighting (see Figure 8.3). In the simulation, we introduce a unit set-point change for output y_1 and zero set-point signal for output y_2 . Without constraints, the responses when using exponential data weighting are almost identical to those from LQR design (not shown here), and exhibit faster set-point responses than those from not using exponential weighting. We also compare the constrained control results with and without exponential data weighting, where we only impose the constraints on the first sample of the control signals. All constraints are satisfied for both cases. Figure 8.4 shows the comparative results. It is seen that the responses are quite different. Again, the output responses from using exponential data weighting are faster than those without exponential data weighting.

Table 8.3. Elements of the first row in K_{lqr} , K_{mpc} with and without exponential data weighting

K_{lqr}	2.7	-11.2	3.9	-3.0	3.1	1.2	-2.9	9.0	3.5	1.1	0.8	-0.6
$K_{mpc}(\alpha = 0.18)$	2.7	-11.2	3.9	-3.0	3.1	1.2	-2.8	8.9	3.5	1.1	0.8	-0.6
$K_{mpc}(\alpha = 0)$	0.4	-1.7	0.6	-0.4	0.5	0.4	0.6	0.7	-0.1	0.3	0.2	-0.2

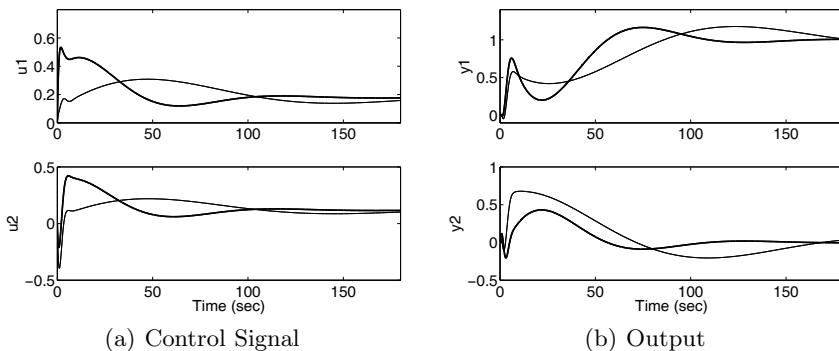


Fig. 8.3. Comparison of CMPC with and without exponential data weighting.
Key: solid-line without exponential data weighting $\alpha = 0$; darker-solid-line with exponential data weighting $\alpha = 0.18$

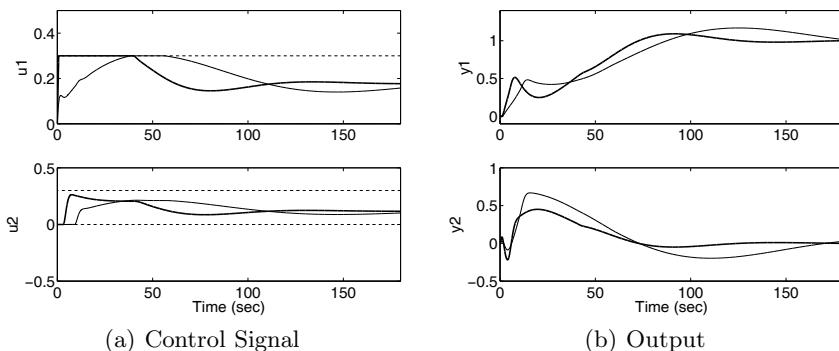


Fig. 8.4. Comparison of CMPC with and without exponential data weighting, in the presence of constraints. Key: solid-line without exponential data weighting $\alpha = 0$; darker-solid-line with exponential data weighting $\alpha = 0.18$

8.7 Summary

This chapter has discussed continuous-time model predictive control with exponential data weighting. In the original design of a continuous-time predictive control system, because of embedded integrator(s) in the model, the prediction horizon is limited to a finite value, and a numerical ill-conditioning problem occurs when the prediction horizon is large. These problems are resolved in this chapter by choosing a cost function with an exponential weight factor $e^{-2\alpha t}$, where $\alpha > 0$:

$$J = \int_0^{T_p} [e^{-2\alpha\tau} x(t_i + \tau \mid t_i)^T Q x(t_i + \tau \mid t_i) + e^{-2\alpha\tau} \dot{u}(\tau)^T R \dot{u}(\tau)] d\tau, \quad (8.53)$$

subject to

$$\dot{x}(t_i + \tau | t_i) = Ax(t_i + \tau | t_i) + Bu(\tau).$$

With the exponential weight, the optimization problem at time t_i is solved based on a pair of transformed variables $x_\alpha(t_i + \tau | t_i)$ and $u_\alpha(\tau)$ by minimizing

$$J = \int_0^{T_p} [x_\alpha(t_i + \tau | t_i)^T Q x_\alpha(t_i + \tau | t_i) + \dot{u}_\alpha(\tau)^T R \dot{u}_\alpha(\tau)] d\tau, \quad (8.54)$$

subject to

$$\dot{x}_\alpha(t_i + \tau | t_i) = (A - \alpha I)x_\alpha(t_i + \tau | t_i) + Bu_\alpha(\tau),$$

where the transformed variables are defined by

$$x_\alpha(t_i + \tau | t_i) = e^{-\alpha\tau}x(t_i + \tau | t_i); \quad u_\alpha(\tau) = e^{-\alpha\tau}u(\tau).$$

The initial conditions are identical when $\tau = 0$. The central idea is that when the system matrix A contains eigenvalues on the imaginary axis or on the right-half of the complex plane, by choosing a suitable $\alpha > 0$ such that the eigenvalues of the modified system matrix $A - \alpha I$ are all strictly on the left-half complex plane, then the model used for prediction is stable, and a sufficiently large prediction horizon can be used in the design. As a consequence, the numerical conditioning problem is overcome. Without any modification on the pair of weight matrices Q, R , the solution does not guarantee exponential decay of the original variable $x(t_i + \tau | t_i)$ within the optimization window. To resolve this issue, a simple modification of the weight matrix Q is proposed. Choosing $Q_\alpha = Q + 2\alpha P$, $\alpha > 0$, R unchanged, the optimal control $\dot{u}(\tau)$ is obtained by minimizing

$$J = \int_0^{T_p} [x_\alpha(t_i + \tau | t_i)^T Q_\alpha x_\alpha(t_i + \tau | t_i) + \dot{u}_\alpha(\tau)^T R \dot{u}_\alpha(\tau)] d\tau, \quad (8.55)$$

subject to

$$\dot{x}_\alpha(t_i + \tau | t_i) = (A - \alpha I)x_\alpha(t_i + \tau | t_i) + Bu_\alpha(\tau),$$

where P is the solution of the steady-state Riccati equation:

$$PA + A^T P - PBR^{-1}B^T P + Q = 0.$$

In fact, the optimal solution with the exponentially weighted cost function is identical to the original optimal control solution without exponential weighting, when the prediction horizon is sufficiently large. The proposed approach is not only numerically sound, but also allows the use of a sufficiently large prediction horizon to guarantee asymptotic stability.

To introduce a prescribed degree of stability β in the predictive control system such that all the closed-loop eigenvalues are on the left of the line $s = -\beta$ in the complex plane, we only need to choose Q_α as

$$Q_\alpha = Q + 2(\alpha + \beta)P,$$

and R unchanged and minimize the cost function J given by (8.55) with transformed variables x_α and u_α , where P is the solution of the steady-state Riccati equation:

$$P(A + \beta I) + (A + \beta I)^T P - PBR^{-1}B^T P + Q = 0.$$

When imposing constraints, all the constraints are transformed and expressed using the exponentially weighted variables.

Problems

8.1. A mathematical model for an inverted pendulum is described by the Laplace transfer function:

$$G(s) = \frac{-K_i}{s^2 - a^2}, \quad (8.56)$$

where the input to the inverted pendulum is external force, and output is angle θ (rad) (see Figure 8.5). The parameters in the model are $K_i = 0.01$ and $a = 3$.

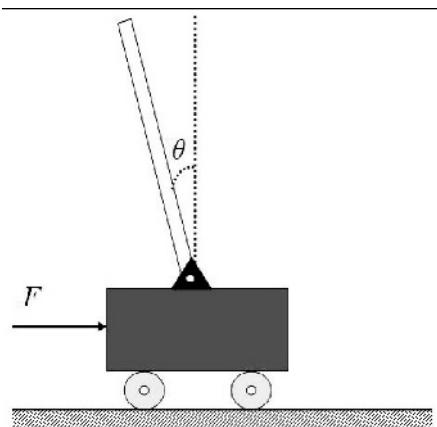


Fig. 8.5. Schematic diagram for an inverted pendulum

Since the inverted pendulum is an unstable system with one pole on the right-half of the complex plane, the choice of the prediction horizon needs careful consideration without using exponential data weighting. The design parameters are $N = 4$, $p = 3.3$, $Q = C^T C$ (C is the output matrix of the augmented model) and $R = 0.1$. Design a continuous-time predictive control system with a final prediction horizon that will bring the angle θ as close as possible to 0° in the presence of input step disturbance.

1. Show that the Hessian matrix Ω in the cost function

$$J = \eta^T \Omega \eta + 2\eta^T \Psi$$

is numerically ill-conditioned by examining its condition number with respect to an increasing prediction horizon T_p .

2. Demonstrate that this numerical sensitivity causes the variations of the closed-loop feedback control gain K_{mpc} and the closed-loop pole locations.

- 8.2.** Continue from Problem 8.1 and use exponential data weighting in the design of predictive control for this inverted pendulum.

1. Choose the exponential weight factor $\alpha = 3.8$ that is greater than the unstable pole, and modify the weight matrix Q_α according to

$$Q_\alpha = Q + 2\alpha P; PA + A^T P - PBR^{-1}B^T P + Q = 0,$$

where A and B are the matrices in the augmented state-space model.

2. With exponential data weighting, examine the elements of Ω and Ψ matrices in the cost function

$$J = \eta^T \Omega \eta + 2\eta^T \Psi,$$

as functions of prediction horizon T_p and demonstrate graphically that the diagonal elements in Ω converge to constants as T_p increases.

3. For a large T_p , compute the closed-loop feedback control gain K_{mpc} and the closed-loop poles with the predictive control system. Compare them with K_{lqr} and the LQR closed-loop poles (use the MATLAB `lqr` function for this computation).
4. If K_{mpc} and K_{lqr} are not sufficiently close to your expectation, increase the number of terms N in the Laguerre functions to improve the accuracy.

- 8.3.** A continuous-time system has three inputs and two outputs described by the Laplace transfer function

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) & G_{13}(s) \\ G_{21}(s) & G_{22}(s) & G_{23}(s) \end{bmatrix}, \quad (8.57)$$

where $G_{11}(s) = \frac{1}{(s+1)^3}$, $G_{12}(s) = \frac{0.1}{0.1s+1}$, $G_{13}(s) = \frac{-0.8}{s+4}$, $G_{21}(s) = \frac{0.01}{s+1}$, $G_{22}(s) = \frac{(-3s+1)}{(10s+1)(3s+1)}$, $G_{23}(s) = \frac{-0.4}{0.3s+1}$.

1. Find the state-space model and augment it with integrators.
2. Choose $Q = C^T C$ and $R = I$, $p_1 = p_2 = p_3 = 0.8$, and $N_1 = N_2 = N_3 = 3$ as the design parameters. Find the matrices Ω and Ψ in the cost function of the predictive control J , where J is expressed as

$$J = \eta^T \Omega \eta + 2\eta^T \Psi,$$

such that the closed-loop eigenvalues of the predictive control system are positioned on the left of a line $s = -1$ in the complex plane. Hint: you need to solve the following Riccati equation to find P matrix

$$P(A + I) + (A + I)^T P - PBR^{-1}B^T P + Q = 0,$$

with exponential weight factor $\alpha > 0$ (say $\alpha = 0.5$), Q is modified to $Q_\alpha = Q + 2(\alpha + 1)P$. The solution of the Riccati equation is performed using MATLAB lqr function.

3. We can also design an observer with the observer poles to be constrained. For instance, if we want to position the observer poles on the left of a line $s = -\gamma$ ($\gamma > 0$) in the complex plane, we choose $Q_{ob} = I$ and $Rob = 0.1I$, and then modify A with $A + \gamma I$. The MATLAB script for doing this is

```
K_ob=lqr((A+gamma*eye(n,n))',C',Qob,Rob);
```

where n is the dimension of A matrix. Find the observer K_{ob} such that the closed-loop observer poles are on the left of a line $s = -2$ in the complex plane.

- 8.4.** Consider the problems of using a prescribed degree of stability to improve the robustness of a predictive control system. Assume that the open-loop system is described by a transfer function

$$G(s) = \frac{K}{(10s + 1)^2(s - 0.3)}.$$

1. Assuming $K = 1$, design a predictive control system with prescribed degree of stability $\beta = 0.4$ (*i.e.*, all closed-loop eigenvalues are on the left of a line $s = -0.4$ in the complex plane). The remaining design parameters are specified as $N = 6$, $p = 0.6$, $Q = C^T C$, $R = 1$ and $T_p = 35$. Parameter α is chosen as $\alpha = 0.38$ to be greater than the magnitude of the unstable pole. An observer is used in the implementation. The weight matrices for the observer are $Q_{ob} = I$ and $Rob = 0.0001$.
2. Construct the closed-loop predictive control system, respectively, with $K = 0.8, 0.9, 1, 1.2, 1.4$ and 1.6 , and calculate the closed-loop eigenvalues with the variations of parameter K , show that the closed-loop system is stable for this range of parameters.
3. Repeat the design with $\beta = 0$, but the other design parameters remaining the same. Show that the closed-loop system is only stable with respect to the changes of K between 0.9 and 1.1 . Present your comparative results using a tabulation of closed-loop eigenvalues, and comment on your findings.

Classical MPC Systems in State-space Formulation

9.1 Introduction

Dynamic matrix control (DMC) and generalized predictive control (GPC) are two classes of predictive control systems that have found application in many areas. This chapter will link the predictive control systems designed using the framework of state space to the classical predictive control systems. One of the common features of the classical predictive control systems is the direct utilization of plant input and output signals in the closed-loop feedback control, hence avoiding observers in the implementation. The key to the link is to revise the classical predictive control schemes using a special class of state-space formulations, where the state variables are chosen to be identical to the feedback variables that have been used in the classical predictive control systems. An example of a state-space formulation of GPC is the work by Ordys and Clarke (1993). Once the state-space model is formulated, the framework from the previous chapters is naturally extended to the classical predictive control systems, preserving all the advantages of a state-space design, including stability analysis, exponential data weighting and LQR equivalence. In addition, because of the direct use of plant input and output signals in the implementation, the predictive controller can be represented in a transfer function form, allowing direct frequency response analysis of the system to obtain critical information, such as gain and phase margins.

More specifically, in Section 9.2, we will discuss GPC in state-space formulation, where we select the same feedback variables and the same cost function in the design. It is seen that the design methodology becomes quite straightforward once the state-space formulation is used. In GPC formulation, the state variables are the output and delayed output signals, the system matrices (A, B, C) need to be modified to incorporate integrators, also its implementation involves an equivalent set-point filter. Instead, the formulation we used in the previous chapters (see Chapters 1 to 4) offers a simplified approach, and in Section 9.3, we present this alternative approach to GPC, completed with transfer function analysis, root-locus analysis and Nyquist plots. The sim-

plicity of the algorithms remains when they are extended to multi-input and multi-output systems (see Section 9.4). In Sections 9.5 and 9.6 we discuss the continuous-time counterpart of the model predictive control using the NMSS formulation. Different from discrete time, in the continuous-time design, an implementation filter is required, and the poles of the filter become part of the desired closed-loop poles when we choose to optimize the output errors in the design. In Section 9.7, we discuss predictive control using impulse response models, which is a special case of general transfer function representation of the plant. Along similar lines, discrete-time Laguerre models could offer an alternative to impulse response models.

9.2 Generalized Predictive Control in State-space Formulation

In the previous chapters, we assumed that state-space models in both continuous time and discrete time are controllable and observable. Controllability is required in order for the design of MPC to access all dynamic modes in the plant, and observability is required for the design of observer to access all plant dynamic modes through measurement of input and output signals.

9.2.1 Special Class of Discrete-time State-space Structures

In this chapter, the model predictive control systems are described using a special class of state-space models. This special formulation uses plant input and output variables as its state variables. Hence, in the implementation of the predictive control system, an observer is not required. A simple example of this class of state-space models is shown below.

Example 9.1. Suppose that a discrete-time transfer function model is given as

$$G(z) = \frac{(z - 0.1)z^{-2}}{(z - 0.6)(z - 0.8)}. \quad (9.1)$$

Let $u(k)$ and $y(k)$ denote the input and output signals, respectively. Choose

$$x(k) = [y(k) \ y(k-1) \ u(k-1) \ u(k-2) \ u(k-3)]^T.$$

Show that the state-space representation of the transfer function model is a non-minimal realization and the model is controllable, but not observable.

Solution. This is a fourth-order discrete-time system with 4 poles ($z = 0$, $z = 0$, $z = 0.6$ and $z = 0.8$). A minimal state-space realization of this transfer function should have four state variables.

Using the forward shift operator, the difference equation that relates the input signal $u(k)$ to the output signal $y(k)$ through the transfer function (9.1) is expressed as

$$y(k+4) - 1.4y(k+3) + 0.48y(k+2) = u(k+1) - 0.1u(k). \quad (9.2)$$

This is the forward shift version of the difference equation. However, at sampling instant k , future information on u and y is required. To avoid this, (9.2) is multiplied with a backward shift q^{-3} , leading to

$$y(k+1) = 1.4y(k) - 0.48y(k-1) + u(k-2) - 0.1u(k-3). \quad (9.3)$$

Choosing the state variable vector as

$$x(k) = [y(k) \ y(k-1) \ u(k-1) \ u(k-2) \ u(k-3)]^T,$$

then, the state-space model is

$$\begin{bmatrix} y(k+1) \\ y(k) \\ u(k) \\ u(k-1) \\ u(k-2) \end{bmatrix} = \begin{bmatrix} 1.4 & -0.48 & 0 & 1 & -0.1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y(k) \\ y(k-1) \\ u(k-1) \\ u(k-2) \\ u(k-3) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} u(k)$$

$$y(k) = [1 \ 0 \ 0 \ 0 \ 0] x(k). \quad (9.4)$$

It is seen that the number of state variable in this special realization is five, instead of four which is the number corresponding to a minimal realization. By using the MATLAB® script listed as below, controllability and observability can be checked, where (A, B, C) are the system matrices in (9.4).

```
AB=ctrb(A,B);
rank(AB)
CA=ctrb(A',C');
rank(CA)
```

The numerical results are: the rank of the controllability matrix is 5; and the rank of the observability matrix is 4. Hence, the state-space model is controllable, but not observable. The eigenvalues of the system matrix are 0.6; 0.8; 0; 0; 0, where the first four eigenvalues are from the poles of the transfer function (*i.e.* the system) while the last eigenvalue at 0 is from the extra state variable used in the special realization.

Young and his colleagues (see Young *et al.*, 1987, Wang and Young, 1988, Chotai *et al.*, 1998) have termed this special realization as non-minimal state-space realization (NMSS). The terminology of non-minimal state-space realization comes from the opposite of minimal realization. In Kailath (1980), a realization (A_m, B_m, C_m) is minimal if it has the smallest number of state variables among all realizations having the same transfer function $G_m(z) = C_m(zI - A_m)^{-1}B_m$. For a transfer function $G_m(z)$, there could be many minimal realizations, but they share the same important properties. For instance, a realization is minimal if and only if the denominator $A(z) = \det(zI - A_m)$

and the numerator $B(z) = C_m \text{Adj}(zI - A_m)B_m$ are relatively prime, where $\text{Adj}(\cdot)$ denotes the adjugate matrix with $(zI - A_m)^{-1} = \frac{\text{Adj}(zI - A_m)}{\det(zI - A_m)}$. For a single-input and single-output system, the smallest number of state variables is equal to the order of denominator, which is the number of state variables for a minimal realization. A realization (A_m, B_m, C_m) is minimal if and only if the pair (A_m, B_m) is controllable and the pair (C_m, A_m) is observable.

In a general term, a state-space model which does not have a minimal realization is a non-minimal state-space (NMSS) model. However, what is of interest to us is this special class of non-minimal state space realizations that will utilize the input and output signals as the state-space variables, hence, rendering the state variables measurable. In addition, the state variables in the feedback configuration are similar to the classical predictive control systems such as GPC and DMC.

We will present a state-space formulation of GPC using the special class of state-space models. Generalized predictive control (GPC) is designed using a transfer function model, with the assumption that the disturbance is a random walk. Namely the disturbance $d(k)$ is represented in the form that

$$d(k) = \frac{1}{1 - q^{-1}} \epsilon(k),$$

where $\epsilon(k)$ is a white noise signal. With this assumption, the incremental control $\Delta u(k)$ is used as the signal to be optimized in the design and an integrator is embedded in the structure of the predictive controller.

The non-minimal state-space model can also deal with disturbances by incorporating disturbance models. With this formulation, an identical predictive control system to GPC can be obtained in the framework of state-space. The following example shows how to incorporate a random walk disturbance into the non-minimal state-space model.

Example 9.2. Suppose that a discrete-time system is described by the dynamic model

$$y(k) = \frac{q^{-1} - 0.5q^{-2}}{1 - 1.6q^{-1} + 0.64q^{-2}} u(k) + \frac{1 - 1.6q^{-1} + 0.64q^{-2}}{1 - q^{-1}} \epsilon(k-1), \quad (9.5)$$

where q^{-1} is the backward shift operator and $\epsilon(k)$ is a white noise sequence; $y(k)$ and $u(k)$ are the output and input signal. The disturbance to the system contains an integrator. Find the state-space expression of the dynamic system with input signal $\Delta u(k)$ and white noise input disturbance $\epsilon(k)$. Analyze the eigenvalues and check controllability and observability of the model.

Solution. With the inverse filtering by the disturbance model, (9.5) becomes

$$(1 - 1.6q^{-1} + 0.64q^{-2})(1 - q^{-1})y(k) = (q^{-1} - 0.5q^{-2})(1 - q^{-1})u(k) + \epsilon(k-1), \quad (9.6)$$

which is

$$(1 - 2.6q^{-1} + 2.24q^{-2} - 0.64q^{-3})y(k) = (q^{-1} - 0.5q^{-2})\Delta u(k) + \epsilon(k-1), \quad (9.7)$$

where $\Delta u(k) = u(k) - u(k-1)$. With $y(k+1)$ as the leading term, the difference equation for (9.7) is

$$y(k+1) = 2.6y(k) - 2.24y(k-1) + 0.64y(k-2) + \Delta u(k) - 0.5\Delta u(k-1) + \epsilon(k). \quad (9.8)$$

Choosing the state variable vector as

$$x(k) = [y(k) \ y(k-1) \ y(k-2) \ \Delta u(k-1)]^T,$$

the state-space model is expressed as

$$\begin{aligned} \begin{bmatrix} y(k+1) \\ y(k) \\ y(k-1) \\ \Delta u(k) \end{bmatrix} &= \begin{bmatrix} 2.6 & -2.24 & 0.64 & -0.5 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y(k) \\ y(k-1) \\ y(k-2) \\ \Delta u(k-1) \end{bmatrix} \\ &\quad + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \Delta u(k) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \epsilon(k) \\ y(k) &= [1 \ 0 \ 0 \ 0 \ 0] x(k). \end{aligned} \quad (9.9)$$

By following the same procedure as in Example 9.1, we find that the NMSS model is controllable, but not observable. The eigenvalues of the system matrix are 0.8, 0.8, 1, 0. The eigenvalue at 1 comes from the incorporation of the integrator, and 0 comes from the additional state variable associated with the input delay.

9.2.2 General NMSS Structure for GPC Design

Once the problem is formulated in terms of a state-space model, the predictive control using the receding horizon control principle is solved in terms of the state-space model. The prediction horizon can be taken to infinity with exponential data weighting, also a prescribed degree of stability can be incorporated in the design.

In the general discrete-time model, the input and output relationship is described by the equation:

$$F(q^{-1})(1 - q^{-1})y(k) = H(q^{-1})\Delta u(k) + q^{-1}\epsilon(k), \quad (9.10)$$

where the polynomials $F(q^{-1})$ and $H(q^{-1})$ are given in the following forms:

$$\begin{aligned} F(q^{-1}) &= 1 + f_1q^{-1} + f_2q^{-2} + \dots + f_nq^{-n} \\ H(q^{-1}) &= h_1q^{-1} + h_2q^{-2} + \dots + h_nq^{-n}. \end{aligned}$$

Let the polynomial $F(q^{-1})(1 - q^{-1})$ be denoted by

$$F(q^{-1})(1 - q^{-1}) = 1 + \bar{f}_1 q^{-1} + \bar{f}_2 q^{-2} + \dots + \bar{f}_{n+1} q^{-(n+1)}.$$

Then, by choosing the state variable vector as

$$x(k) = [y(k) \ y(k-1) \ \dots \ y(k-n-1) \ \Delta u(k-1) \ \dots \ \Delta u(k-n)]^T,$$

the state-space model with non-minimal realization is expressed as

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) + B_d\epsilon(k) \\ y(k) &= Cx(k), \end{aligned} \quad (9.11)$$

where A is a matrix of $(2n+1) \times (2n+1)$; B is a matrix of $(2n+1) \times 1$ and C is $1 \times (2n+1)$. More specifically,

$$A = \begin{bmatrix} -\bar{f}_1 & -\bar{f}_2 & \dots & -\bar{f}_{n-1} & -\bar{f}_n & h_2 & \dots & h_{n-1} & h_n \\ 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}; \quad B = \begin{bmatrix} h_1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$C = [1 \ 0 \ 0 \ \dots \ 0 \ 0 \ 0 \ 0]; \quad B_d^T = [1 \ 0 \ 0 \ \dots \ 0 \ 0 \ 0 \ 0].$$

9.2.3 Generalized Predictive Control in State-space Formulation

After the state-space model is formulated, at sample time k_i and for a given initial condition $x(k_i)$, the prediction of the state variables is obtained for the future sampling time m as

$$\begin{aligned} x(k_i + m \mid k_i) &= A^m x(k_i) + \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \eta \\ &\quad + \sum_{i=0}^{m-1} A^{m-i-1} B_d \epsilon(k_i + i \mid k_i), \end{aligned} \quad (9.12)$$

where the parameter vector η comprises N Laguerre coefficients:

$$\eta = [c_1 \ c_2 \ \dots \ c_N]^T.$$

Note that the noise $\epsilon(k)$ is assumed to be a white noise sequence with zero mean and the optimal prediction of a zero-mean white noise sequence is a

zero sequence. Thus, effectively, the noise term vanishes from the prediction equation (9.12) to yield

$$x(k_i + m \mid k_i) = A^m x(k_i) + \sum_{i=0}^{m-1} A^{m-i-1} B L(i)^T \eta. \quad (9.13)$$

If the set-point signal is zero, for a large prediction horizon N_p , the cost function is chosen as

$$\begin{aligned} J &= \sum_{j=1}^{N_p} \alpha^{-2j} x(k_i + j \mid k_i)^T Q_\alpha x(k_i + j \mid k_i) \\ &\quad + \sum_{j=0}^{N_p} \alpha^{-2j} \Delta u(k_i + j)^T R_\alpha \Delta u(k_i + j), \end{aligned} \quad (9.14)$$

where $Q = C^T C$, $\alpha > 1$, $0 < \lambda < 1$; Q_α and R_α satisfy

$$\gamma = \frac{\lambda}{\alpha} \quad (9.15)$$

$$Q_\alpha = \gamma^2 Q + (1 - \gamma^2) P_\infty \quad (9.16)$$

$$R_\alpha = \gamma^2 R. \quad (9.17)$$

Here, P_∞ is the solution of the Riccati equation:

$$\frac{A^T}{\lambda} [P_\infty - P_\infty \frac{B}{\lambda} (R + \frac{B^T}{\lambda} P_\infty \frac{B}{\lambda})^{-1} \frac{B^T}{\lambda} P_\infty] \frac{A}{\lambda} + Q - P_\infty = 0,$$

where the matrices $\hat{A} = \alpha^{-1} A$ and $\hat{B} = \alpha^{-1} B$. The optimal solution $\Delta u(\cdot)$ converges to the LQR solution with a prescribed degree of stability $0 < \lambda < 1$, as discussed in Chapter 4. When constraints are imposed, the predictive control problem is converted to an optimization of the objective function J subject to a set of linear inequality constraints.

If the set-point signal $r(k) \neq 0$, then the output variables, $y(k)$, $y(k-1)$, ..., are replaced by the error signals $y(k) - r(k)$, $y(k-1) - r(k-1)$, ..., in the state variable vector $x(k)$.¹ The form of the cost function remains unchanged. Because of receding horizon control principle, the information for set-point following is translated into initial conditions of the state variables at the beginning of the optimization window.

Example 9.3. Use the same system as given in Example 9.2, where

$$y(k) = \frac{q^{-1} - 0.5q^{-2}}{1 - 1.6q^{-1} + 0.64q^{-2}} u(k) + \frac{1 - 1.6q^{-1} + 0.64q^{-2}}{1 - q^{-1}} \epsilon(k-1). \quad (9.18)$$

¹ The error signals are defined by subtracting the set-point signals from the outputs and ensuring that negative state feedback is achieved.

Choosing $Q = C^T C$ and $R = 1$, show that the predictive control system with $\alpha = 1.2$, $N_p = 46$, $a = 0.3$ and $N = 3$ is a close approximation to the LQR design. Simulate the predictive control system with unit set-point change and demonstrate rejection of a constant input disturbance with unit amplitude. Measurement noise with standard deviation of 0.01 is added to the output in the simulation.

Solution. The feedback control gain vector using the MATLAB function ‘`dlqr`’ is

$$K = [1.7626 \ -1.8848 \ 0.5898 \ -0.4608].$$

The predictive control is designed by minimizing the cost function:

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(k_i),$$

where the data matrices Ω and Ψ are calculated using the results presented in Chapters 3 and 4 as

$$\Omega = \begin{bmatrix} 48.682 & 46.861 & 44.363 \\ 46.861 & 48.682 & 46.861 \\ 44.363 & 46.861 & 48.682 \end{bmatrix}; \Psi = \begin{bmatrix} 96.330 & -138.411 & 51.325 & -40.098 \\ 93.373 & -137.046 & 51.376 & -40.138 \\ 88.103 & -131.483 & 49.799 & -38.905 \end{bmatrix}.$$

Thus, the feedback gain vector via the computation of predictive control is

$$K_{mpc} = L(0)^T \Omega^{-1} \Psi = [1.7783 \ -1.8723 \ 0.5682 \ -0.4439].$$

For this particular example, if the number of terms, N , in the Laguerre functions increases from 3 to 4, the predictive control gain vector converges to the underlying discrete-time LQR gain vector, where the computational result shows that

$$K_{mpc} = [1.7626 \ -1.8848 \ 0.5898 \ -0.4608].$$

Figure 9.1 shows the control signal and output response for the generalized predictive control (GPC) system using the state-space formulation, in which the unit set-point signal entered the system at $k = 0$ and the unit constant input disturbance entered the system at $k = 40$. A white noise sequence was added to the output to simulate measurement noise. From Figure 9.1b, we can see that the output y has followed the set-point change without steady-state error and the unit constant input disturbance has been rejected rapidly.

Note that in this formulation the exponential data weighting is used to obtain a numerically well-conditioned solution. The condition number of the Hessian matrix Ω is 152. In contrast, if we set $\alpha = 1$, the data matrices Ω and Ψ would take the values:

$$\Omega = \begin{bmatrix} 1.0245 & 0.9946 & 0.9595 \\ 0.9946 & 0.9707 & 0.9408 \\ 0.9595 & 0.9408 & 0.9168 \end{bmatrix} \times 10^4$$

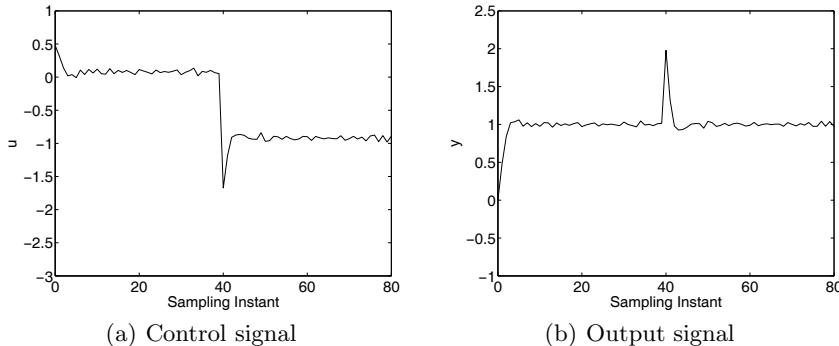


Fig. 9.1. Generalized predictive control using state-space formulation

$$\Psi = \begin{bmatrix} 1.5107 & -2.3981 & 0.9531 & -0.7446 \\ 1.4651 & -2.3293 & 0.9268 & -0.7240 \\ 1.4120 & -2.2480 & 0.8953 & -0.6995 \end{bmatrix} \times 10^4.$$

The condition number of Ω is $\kappa(\Omega) = 12864$, which clearly indicates that the Hessian matrix is ill-conditioned.

9.3 Alternative Formulation to GPC

The non-minimal state-space model was derived based on the input-output model (9.10), where the input to the model is Δu and the output from the model is y . This follows the GPC formulation using output y as the feedback variable. Alternatively, the approaches introduced in the previous chapters used the incremental of the state variables as part of the feedback signals, namely Δx_m . The corresponding variables here are the Δy variables. Both approaches yield a similar closed-loop performance as long as the cost functions are the same. However, there are a few advantages when using $\Delta y(k)$ as part of the feedback signals. Firstly, when using $\Delta y(k)$ as part of the feedback signal, then (9.10) becomes

$$F(q^{-1})\Delta y(k) = H(q^{-1})\Delta u(k) + \epsilon(k). \quad (9.19)$$

Thus the coefficients in the state-space model are directly related to the original transfer function model. Also, the steady-state values of $\Delta y(k)$ are zero for constant reference signals and disturbances, therefore ignored, leading to convenience in implementation. Both advantages are even more significant for a multi-input and multi-output system.

9.3.1 Alternative Formulation for SISO Systems

We assume that a z -transfer function is given by

$$G_m(z) = \frac{H(z)}{F(z)} z^{-d}, \quad (9.20)$$

where the polynomial $F(z)$ and $H(z)$ are defined as

$$\begin{aligned} F(z) &= z^n + f_1 z^{n-1} + \dots + f_n \\ H(z) &= h_1 z^{n-1} + h_2 z^{n-2} + \dots + h_n. \end{aligned}$$

As the number of time delays is counted as part of the model order in the discrete-time system, the minimal realization of a state space model has the number of state variables equal to $n+d$. However, in a non-minimal realization by taking the state variables as

$$x_m(k)^T = [y(k) \ y(k-1) \ \dots \ y(k-n+1) \ u(k-1) \ \dots \ u(k-n-d+1)],$$

the number of state variables is equal to $2n+d-1$.² With this choice of state variables, we obtain the particular state-space realization as

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k). \end{aligned} \quad (9.21)$$

The matrices A_m , B_m , C_m are defined as the block matrices

$$A_m = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}; \quad B_m = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}; \quad C_m = \begin{bmatrix} C_1 & C_2 \end{bmatrix},$$

where the matrix A_1 has dimension $n \times n$, A_4 has the dimension $(n+d-1) \times (n+d-1)$, A_2 has the dimension $n \times (n+d-1)$, and A_3 is a zero matrix and has the dimension $(n+d-1) \times n$. More specifically, for $d \neq 0$,

$$\begin{aligned} A_1 &= \begin{bmatrix} -f_1 & -f_2 & \dots & -f_{n-1} & -f_n \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}; \quad A_2 = \begin{bmatrix} 0 & \dots & h_1 & \dots & h_n \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \\ A_4 &= \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}, \end{aligned}$$

B_1 has the dimension $n \times 1$ and B_2 has the dimension $(n+d-1) \times 1$ with the following forms:

² Note that in the case that $n = 1$, the number of state variables in the NMSS is equal to the number of states in the minimal realization.

$$B_1 = [0 \ 0 \dots 0]^T; B_2 = [1 \ 0 \dots 0]^T,$$

C_2 has the dimension $1 \times (n+d-1)$ and C_1 has the dimension $1 \times n$ with the form: $C_1 = [1 \ 0 \dots 0]$. When $d=0$, there are differences in the matrices A_2 and B_1 , where

$$A_2 = \begin{bmatrix} h_2 & h_3 & \dots & h_{n-1} & h_n \\ 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}; B_1 = [h_1 \ 0 \dots 0]^T.$$

With this NMSS model, the design and implementation of predictive control systems that have been discussed in the previous chapters will be applicable. When embedding an integrator into the NMSS model, as before, we obtain the augmented state-space model as

$$\begin{aligned} \overbrace{\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} &= \underbrace{\begin{bmatrix} A_m & o_m^T \\ C_m A_m & 1 \end{bmatrix}}_A \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \underbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}_B \Delta u(k) \\ y(k) &= \underbrace{\begin{bmatrix} o_m & 1 \end{bmatrix}}_C \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}. \end{aligned} \quad (9.22)$$

Because the state variable vector $x(k)$ consists of $\Delta x_m(k)$ and $y(k)$, we can assume its initial condition is zero when starting up the predictive control system. Noting that the majority of the elements in $x(k)$ are the shifted variables from the previous $x(k-1)$, at sampling time k , it is only necessary to update the $\Delta y(k)$, $\Delta u(k-1)$ and the error $y(k) - r(k)$. Thus, the implementation of the predictive control system is quite simple.

9.3.2 Closed-loop Poles of the Predictive Control System

Because the matrix A_m has the block upper triangular structure with A_3 being a zero matrix, the eigenvalues of A_m comprise the eigenvalues of A_1 and the eigenvalues of A_4 . To calculate the eigenvalues of A_1 , we evaluate $\det(\lambda I - A_1)$, by taking advantage of its special structure, which is

$$\det(\lambda I - A_1) = \lambda^n + f_1 \lambda^{n-1} + \dots + f_n = 0.$$

This gives the n poles that are identical to the poles of the transfer function model $\frac{H(z)}{F(z)}$. Note that the matrix A_4 is a special matrix, and has the characteristic equation,

$$\det(\lambda I - A_4) = \lambda^{n+d-1} = 0,$$

whose solutions give the $(n+d-1)$ poles located on the origin of the complex plane. Here, the $n+d-1$ poles include the poles from the time delay d and the $n-1$ poles from the non-minimal state space realization.

What is particularly of interest to us is that the $n + d - 1$ poles at $\lambda = 0$ are part of the closed-loop poles and cannot be changed by the predictive control system, if we select the weight matrix $Q = C^T C$. The simplest way to illustrate this is to examine the closed-loop eigenvalues of the predictive control through the root locus of DLQR. Assuming that the cost function is, $N_p \rightarrow \infty$,

$$\begin{aligned} J = & \sum_{j=0}^{N_p} x(k_i + j \mid k_i)^T Q x(k_i + j \mid k_i) \\ & + \sum_{j=0}^{N_p} \Delta u(k_i + j)^T R \Delta u(k_i + j), \end{aligned} \quad (9.23)$$

where the weights are $R = r_w$, and $Q = C^T C$, then, the closed-loop eigenvalues are the inside-the-unit-circle zeros of the equation,

$$\rho = 1 + \frac{1}{r_w} \frac{G_{nmss}(z)G_{nmss}(z^{-1})}{(z-1)(z^{-1}-1)} = 0, \quad (9.24)$$

where $G_{nmss} = C_m(zI - A_m)^{-1}B_m$. By going through the matrix computation, we will obtain the transfer function

$$G_{nmss} = C_m(zI - A_m)^{-1}B_m = \frac{H(z)z^{n-1}}{F(z)z^{n-1}}z^{-d}. \quad (9.25)$$

With $(n - 1)$ pole and zero cancellation at the location $z = 0$, the original transfer function $G_m(z)$ returns. However, in order to find the locations of the closed-loop eigenvalues, substituting (9.25) into (9.24), we obtain

$$\rho(z) = \frac{z^{n-1}z^{-(n-1)} [r_w F(z)F(z^{-1})(z-1)(z^{-1}-1)z^d z^{-d} + H(z)H(z^{-1})]}{r_w F(z)F(z^{-1})(z-1)(z^{-1}-1)z^d z^{-d} z^{n-1} z^{-(n-1)}}. \quad (9.26)$$

The closed-loop eigenvalues of the predictive control system are the inside-the-unit-circle zeros of the equation $\rho(z) = 0$.

There are two comments here. From (9.26), one set of the inside-the-unit-circle zeros of the equation are the $n - 1$ zeros on the origin of the complex plane that are from the non-minimal state space structure, which cannot be changed by the predictive controller if we select $Q = C^T C$. In addition, the number of plant's delay samples becomes the poles of the model, which cannot be changed by closed-loop feedback control. Therefore, in total, there are $d + n - 1$ eigenvalues on the origin of the complex plane, which cannot be changed by the predictive controller. The rest of the closed-loop eigenvalues are determined by the inside-unit-circle zeros of the equation

$$\bar{\rho}(z) = \frac{r_w F(z)F(z^{-1})(z-1)(z^{-1}-1) + H(z)H(z^{-1})}{r_w F(z)F(z^{-1})(z-1)(z^{-1}-1)}. \quad (9.27)$$

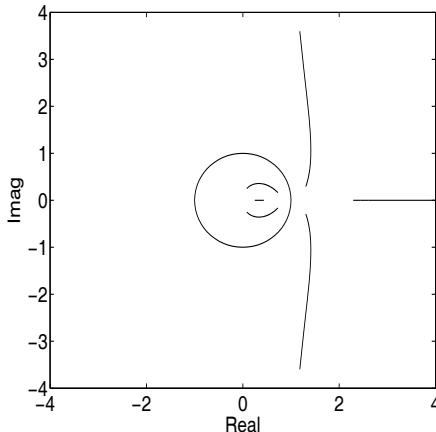


Fig. 9.2. Dual root locus of predictive control system

Example 9.4. Suppose that a discrete-time system is described by the transfer function

$$G_m(z) = \frac{0.5z - 0.1}{z^2 - 1.8z + 0.6} z^{-3}.$$

Determine the eigenvalues of the closed-loop predictive control system with integral action as function of the weight r_w by examining the zeros of (9.27), where we assume that NMSS structure is used, and that $Q = C^T C$.

Solution. Essentially, the polynomial equation needed for determining the closed-loop eigenvalues is

$$r_w(z^2 + f_1z + f_2)(1 + f_1z + f_2z^2)(z - 1)(1 - z) + z^2(h_1z + h_2)(h_1 + h_2z) = 0.$$

This is a sixth-order polynomial which has six zeros. When r_w varies from 0.01 to 20 with increment of 0.01, we compute the zeros of the polynomial equation for each r_w . The dual root locus is shown in Figure 9.2. It is seen that there are three branches inside the unit circle and another three branches outside the unit circle. The three branches outside the unit circle are the inverse of the three branches inside the unit circle. The root locus of the predictive control system are the three branches inside the unit circle. There are another four closed-loop poles ($= n + d - 1 = 4$) at the origin of the complex plane which cannot be changed by the control. We can use the root locus to determine the r_w according to the closed-loop pole locations, and we can also use the root locus to determine the scaling factor of the Laguerre function a . For instance, once we know the dominant closed-loop poles, we can choose a to be close to these dominant poles so the number of terms N , required for optimal control, is reduced.

Let us consider the case when $r_w = 20$. From the root locus, the closed-loop poles are at $0.7275 \pm j0.1633$, 0.4364 , 0 , 0 , 0 , 0 . By choosing $a = 0.7$,

$N = 6$, and $N_p = 38$, the exponential weight $\alpha = 1.2$, we calculate the Ω and Ψ matrices, leading to the feedback predictive control gain matrix:

$$K_{mpc} = [6.1523 \ -2.6683 \ 0.9086 \ 1.3351 \ 1.9148 \ -0.4447 \ 0.1423],$$

which then returns the closed-loop eigenvalues of the actual predictive control system $(A - BK_{mpc})$ as

$$[0.7275 \pm j0.1634 \ 0.4365 \ 0 \ 0 \ 0 \ 0].$$

9.3.3 Transfer Function Interpretation

This state-space predictive control system design can also be interpreted in terms of transfer functions. The interpretation will be particularly useful in assessing the nominal performance of the model predictive control such as gain and phase margins via frequency response analysis tools. We investigate the example below.

Example 9.5. Assume that a discrete-time plant is represented by the transfer function model:

$$G(z) = \frac{h_1 z + h_2}{z^2 + f_1 z + f_2} z^{-3}. \quad (9.28)$$

Find the predictive control system configuration in terms of a transfer function form.

Solution. From (9.28), the difference equation that relates the input signal $u(k)$ to the output $y(k)$ is

$$y(k) = -f_1 y(k-1) - f_2 y(k-2) + h_1 u(k-4) + h_2 u(k-5).$$

So, we define

$$x(k) = [\Delta y(k) \ \Delta y(k-1) \ \Delta u(k-1) \ \Delta u(k-2) \ \Delta u(k-3) \ \Delta u(k-4) \ y(k)]^T,$$

which leads to the augmented NMSS model:

$$x(k+1) = Ax(k) + B\Delta u(k); \ y(k) = Cx(k),$$

where the matrices are

$$A = \begin{bmatrix} -f_1 & -f_2 & 0 & 0 & h_1 & h_2 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -f_1 & -f_2 & 0 & 0 & h_1 & h_2 & 1 \end{bmatrix}; \ B = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$C = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1].$$

With receding horizon control, we have

$$\Delta u(k) = -L(0)^T \Omega^{-1} \Psi x(k).$$

When introducing a set-point signal, the element $y(k)$ in the state vector $x(k)$ is replaced by the error signal $e(k) = y(k) - r(k)$. Explicitly, with the predictive controller, without constraints, the difference of the control signal is expressed as

$$\Delta u(k) = -[k_1^y \ k_2^y \ k_1^u \ k_2^u \ k_3^u \ k_4^u \ k^e] \begin{bmatrix} \Delta y(k) \\ \Delta y(k-1) \\ \Delta u(k-1) \\ \Delta u(k-2) \\ \Delta u(k-3) \\ \Delta u(k-4) \\ y(k) - r(k) \end{bmatrix}.$$

By taking the z -transform of this difference equation, we obtain

$$(1 - z^{-1})U(z) = -(k_1^u z^{-1} + k_2^u z^{-2} + k_3^u z^{-3} + k_4^u z^{-4})(1 - z^{-1})U(z) \\ - (k_1^y + k_2^y z^{-1})(1 - z^{-1})Y(z) - k^e(Y(z) - R(z)). \quad (9.29)$$

This gives the z -transform of the control signal

$$U(z) = -\frac{k^e(Y(z) - R(z))}{(1 - z^{-1})(1 + k_1^u z^{-1} + k_2^u z^{-2} + k_3^u z^{-3} + k_4^u z^{-4})} \\ - \frac{(k_1^y + k_2^y z^{-1})Y(z)}{1 + k_1^u z^{-1} + k_2^u z^{-2} + k_3^u z^{-3} + k_4^u z^{-4}}. \quad (9.30)$$

Define the following polynomial functions:

$$L(z) = 1 + k_1^u z^{-1} + k_2^u z^{-2} + k_3^u z^{-3} + k_4^u z^{-4}, \quad P(z) = k_1^y + k_2^y z^{-1}.$$

Figure 9.3 shows the block diagram of the feedback control system structure. It is clearly seen that the predictive control system has integral action and it also has a two-degrees-of-freedom configuration.

Using the structure of the predictive controller in transfer function form, the robustness of the closed-loop predictive control system can be analyzed. A commonly known approach is to assess the stability margins of the control system, such as gain margin and phase margin. The relevant information is valuable in the design of a control system. The Nyquist plot is the frequency response of the open-loop system $G(z)C(z)$, where $C(z) = \frac{k^e}{(1-z^{-1})L(z)} + \frac{P(z)}{L(z)}$ with $z = e^{j\omega}$, $0 \leq \omega \leq \pi$.

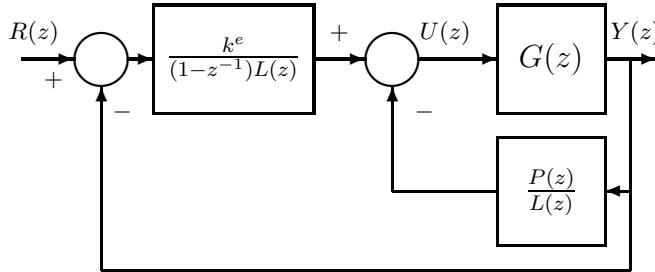


Fig. 9.3. Block diagram of feedback structure using NMSS

Example 9.6. Assume that the plant transfer function is

$$G(z) = \frac{h_1 z + h_2}{z^2 + f_1 z + f_2} z^{-3}$$

where $h_1 = 1$, $h_2 = -0.1$, $f_1 = -1.6$, and $f_2 = 0.68$. The design parameters are $a = 0.6$, $N = 8$, $Q = C^T C$, $\alpha = 1.2$, $N_p = 38$, $R = 10$. Present the Nyquist plot of the predictive control system and evaluate its gain margin.

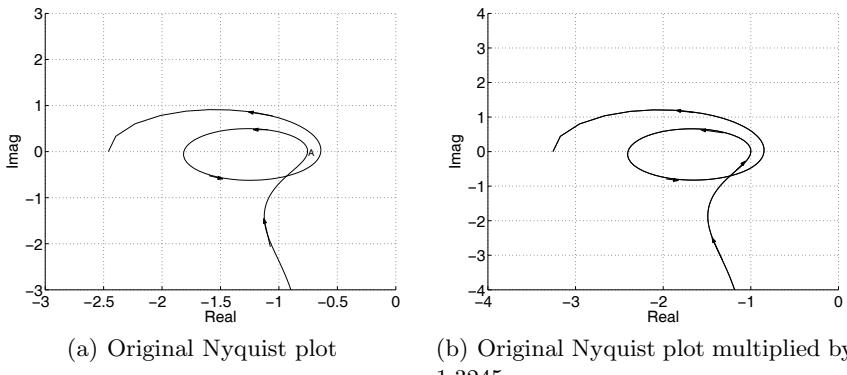
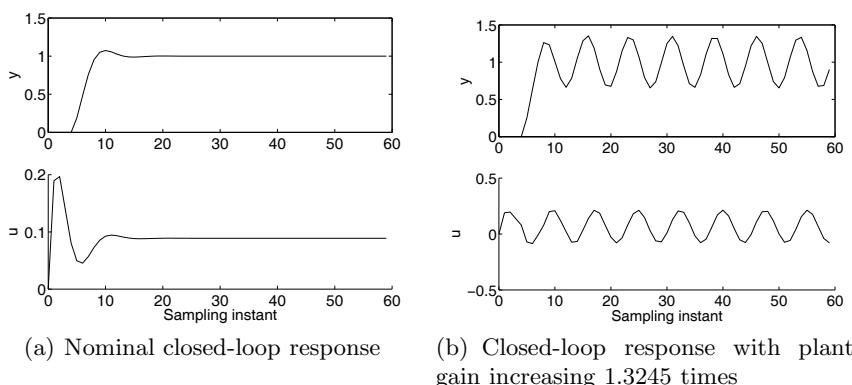
Solution. The gain matrix for the predictive control system is

$$K_{mpc} = [1.77 \ -1.15 \ 0.962 \ 1.273 \ 1.554 \ -0.169 \ 0.189].$$

The corresponding transfer function of the controller has

$$\begin{aligned} L(z) &= 1 + 0.962z^{-1} + 1.273z^{-2} + 1.554z^{-3} - 0.169z^{-4} \\ P(z) &= 1.77 - 1.15z^{-1} \ k^e = 0.189. \end{aligned}$$

The roots of $L(z)$ are at -1.1460 , $0.0422 \pm j1.2141$, 0.1000 , among which are three poles outside the unit circle. Because the plant has all poles inside the unit circle, in conjunction with the information of three unstable poles with the controller, the Nyquist plot will encircle the $(-1, 0)$ point three times in a counter-clock wise manner, if the predictive control system is stable. Figure 9.4a shows that the Nyquist plot of the predictive control system, which indeed has done so. The gain margin is calculated as the inverse of the magnitude of point A ($=0.755$), hence the system has a gain margin of $1/0.755 = 1.3245$. With this information, we know that the predictive control system can tolerate increasing plant gain to a maximum of 1.3245 times the original gain before closed-loop instability occurs. Indeed, when we multiply the original Nyquist plot by 1.3245, the inner branch of the Nyquist plot touches the $(-1, 0)$ point on the complex plane (see Figure 9.4b), which means that the closed-loop predictive control system becomes marginally stable. This result can also be confirmed through closed-loop simulation. Figure 9.5a shows the nominal closed-loop predictive control system response for a unit set-point

**Fig. 9.4.** Nyquist plot of predictive control system**Fig. 9.5.** Closed-loop predictive control system response

change. When multiplying the numerator of the plant model by 1.3245, the closed-loop control system response exhibits sustained oscillation, as shown in Figure 9.5b.

9.4 Extension to MIMO Systems

The extension of the predictive control to multi-input and multi-output systems is based on the formulation of non-minimal state-space models, where the plant input and output variables as the state variables.

9.4.1 MNSS Model for MIMO Systems

Consider that a multi-input and multi-output system is described by the difference equation,

$$\begin{aligned} y(k+1) + F_1 y(k) + F_2 y(k-1) + \dots + F_n y(k-n+1) = \\ H_1 u(k) + H_2 u(k-1) + \dots + H_n u(k-n+1). \end{aligned} \quad (9.31)$$

The state variable vector $x_m(k)$ is chosen as:

$$x_m(k)^T = [y(k)^T \dots y(k-n+1)^T \ u(k-1)^T \dots u(k-n+1)^T].$$

With this choice of state variables, we obtain the particular non-minimal state-space realization as

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x_m(k). \end{aligned} \quad (9.32)$$

The matrices A_m , B_m , and C_m are defined as the block matrices

$$A_m = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}; \quad B_m = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}; \quad C_m = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} -F_1 & -F_2 & \dots & -F_{n-1} & -F_n \\ I & o & \dots & o & o \\ o & I & \dots & o & o \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ o & o & \dots & I & o \end{bmatrix}; \quad A_2 = \begin{bmatrix} H_2 & H_3 & \dots & H_{n-1} & H_n \\ o & o & \dots & o & o \\ \dots & \dots & \dots & \dots & \dots \\ o & o & \dots & o & o \end{bmatrix};$$

$$A_4 = \begin{bmatrix} o & o & \dots & o & o \\ I & o & \dots & o & o \\ o & I & \dots & o & o \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ o & o & \dots & I & o \end{bmatrix}$$

$$B_1 = [H_1 \ o \ \dots \ o]^T; \quad B_2 = [I \ o \ \dots \ o]^T; \quad C_1 = [I \ o \ \dots \ o],$$

where A_3 is a zero matrix, the o symbol denotes the zero matrices with appropriate dimensions. We assume that the number of inputs is equal to the number of outputs for notional simplicity, although the formulation is applicable when the number of inputs is not equal to the number of outputs. When the number of inputs is equal to the number of outputs (assuming m inputs and m outputs), then I is the identity matrix of $m \times m$ and o is the zero matrix of $m \times m$. However, when the number of inputs is not equal to the number of outputs, the dimensions of the zero matrices and identity matrices need to be decided case by case. The following example illustrates how to do this.

Example 9.7. A dynamic system with two inputs and one output is described by the difference equation

$$y(k+1) = -f_1 y(k) - f_2 y(k-1) + [h_1 \ h_2] \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} + [g_1 \ g_2] \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix}.$$

Find the expression of the non-minimal state-space structure in the form of equation (9.32).

Solution. Choosing the state variable vector

$$x_m(k)^T = [y(k) \ u_1(k-1) \ u_2(k-1)],$$

$$\text{then, } \begin{bmatrix} y(k+1) \\ u_1(k) \\ u_2(k) \end{bmatrix} = \begin{bmatrix} -f_1 & -f_2 & g_1 & g_2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y(k) \\ u_1(k-1) \\ u_2(k-1) \end{bmatrix} + \begin{bmatrix} h_1 & h_2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}. \quad (9.33)$$

9.4.2 Case Study of NMSS Predictive Control System

The mathematical model for a glasshouse micro-climate given by Young *et al.* (1994) takes the form:

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} \frac{0.015z^{-1}}{1-0.905z^{-1}} & -0.077z^{-1} \\ \frac{-0.058z^{-1}}{1-0.793z^{-1}} & 0.753z^{-1} \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}, \quad (9.34)$$

where $y_1(k)$ is the air temperature, $y_2(k)$ is the relative humidity of the air, $u_1(k)$ is the fractional valve aperture of the boiler and $u_2(k)$ is the mist spraying system input. We will design and simulate a NMSS-based predictive control system with constraints.

In order to obtain the special form of state-space representation, we need to convert the transfer function model (9.34) to the form of the difference equation by (9.31). Note that the transfer function model in (9.34) is equivalent to

$$G_m(z) = \begin{bmatrix} 1 - .905z^{-1} & 0 \\ 0 & 1 - .793z^{-1} \end{bmatrix}^{-1} \begin{bmatrix} .015z^{-1} & -.077z^{-1}(1 - .905z^{-1}) \\ -.058z^{-1} & .753z^{-1}(1 - .793z^{-1}) \end{bmatrix},$$

which is called a left matrix fraction description (LMFD) (Kalaith, 1980). With this description, the original mathematical model is written as

$$\begin{aligned} & \begin{bmatrix} 1 - 0.905z^{-1} & 0 \\ 0 & 1 - 0.793z^{-1} \end{bmatrix} \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} \\ &= \begin{bmatrix} 0.015z^{-1} & -0.077z^{-1}(1 - 0.905z^{-1}) \\ -0.058z^{-1} & 0.753z^{-1}(1 - 0.793z^{-1}) \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}, \end{aligned}$$

which then leads to the following difference equation

$$\begin{aligned} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} + \begin{bmatrix} -0.905 & 0 \\ 0 & -0.793 \end{bmatrix} \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} \\ &= \begin{bmatrix} 0.015 & -0.077 \\ -0.058 & 0.753 \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} + \begin{bmatrix} 0 & 0.07 \\ 0 & -0.597 \end{bmatrix} \begin{bmatrix} u_1(k-2) \\ u_2(k-2) \end{bmatrix}. \end{aligned} \quad (9.35)$$

The augmented NMSS model is then obtained by choosing the state variable vector as $x(k) = [\Delta y_1(k) \Delta y_2(k) \Delta u_1(k-1) \Delta u_2(k-1) y_1(k) y_2(k)]^T$, input variable vector as $\Delta u(k) = [\Delta u_1(k) \Delta u_2(k)]^T$ and output vector as $y(k) = [y_1(k) y_2(k)]^T$, giving the form

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k). \end{aligned} \quad (9.36)$$

Here,

$$A = \begin{bmatrix} 0.905 & 0 & 0.0 & 0.07 & 0 & 0 \\ 0 & 0.793 & 0 & -0.597 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.905 & 0 & 0.0 & 0.07 & 1 & 0 \\ 0 & 0.793 & 0 & -0.597 & 0 & 1 \end{bmatrix}; \quad B = \begin{bmatrix} 0.015 & -0.077 \\ -0.058 & 0.753 \\ 1 & 0 \\ 0 & 1 \\ 0.015 & -0.077 \\ -0.058 & 0.753 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The air temperature in the glasshouse is set to operate between $15^\circ C$ and $20^\circ C$ respectively ($15 \leq y_1(k) \leq 20$) and the relative humidity of the air is to be maintained at 85% allowing $\pm 0.85\%$ variations ($85 - 0.85 \leq y_2(k) \leq 85 + 0.85$ (percent)), the fractional valve aperture is allowed to vary between 20% and 60% ($20 \leq u_1(k) \leq 60$) and the spray input is allowed to vary between 2 and 18 (mg/s) ($2 \leq u_2(k) \leq 18$). In addition, the rates of changes for the control signals are constrained as: $-1.03 \leq \Delta u_1(k) \leq 1.35$ and $-1.03 \leq \Delta u_2(k) \leq 1.15$. The design parameters in the NMSS-MPC scheme are specified as follows. The Laguerre parameters for both input signals are chosen to be $a_1 = a_2 = 0$, $N_1 = N_2 = 4$; the predictive horizon for the air temperature is chosen as 200 samples and the prediction horizon for the relative humidity is chosen as 100 samples. The weight matrix on the error signal is the identity matrix ($Q = C^T C$) and on the control signal is also the identity matrix ($R = I$).

There are three cases being considered in the simulation studies. In all three cases, the set-point signal r_1 is a square wave, whose amplitude varies between $15^\circ C$ and $20^\circ C$ and the set-point signal r_2 is a constant.

Case A

A white noise sequence with standard deviation of 0.09 is added to the relative humidity of the air to simulate measurement noise. Figure 9.6a and 9.6b show

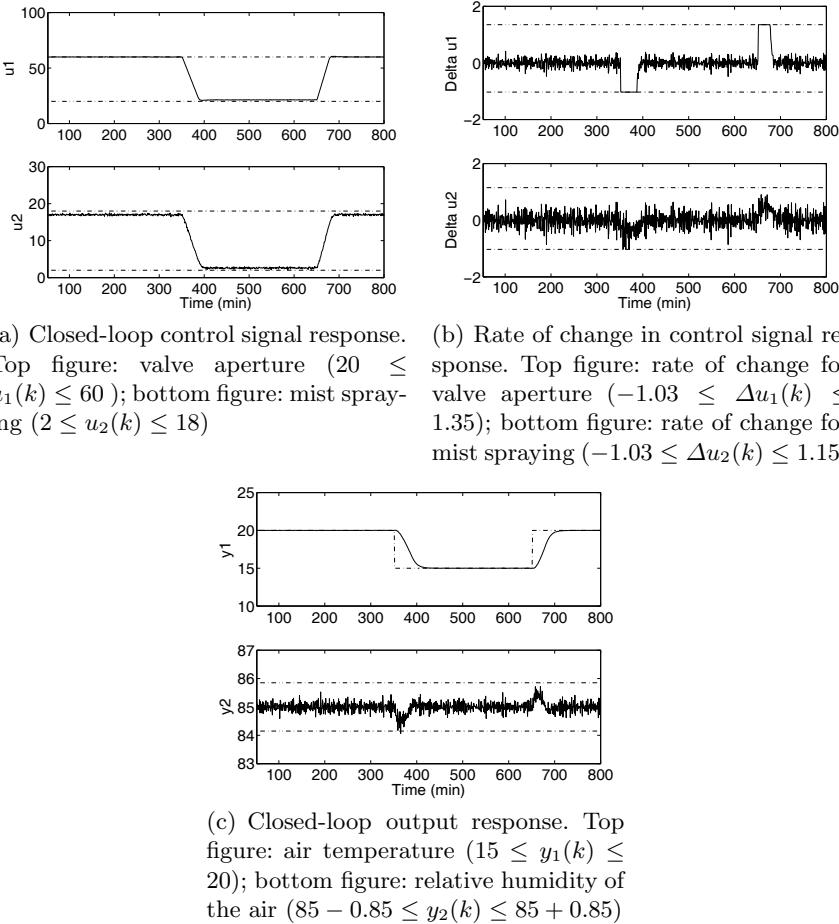


Fig. 9.6. Case A. Measurement noise added to the output

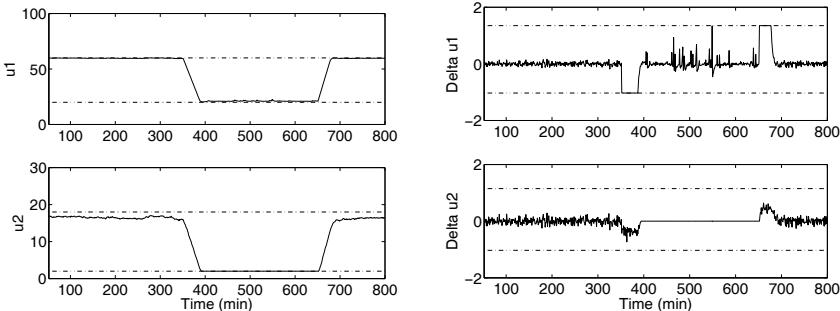
the control signal responses and constrained rates of change on both control signals, respectively, while Figure 9.6c shows the constrained output responses. The figures show that all the control objectives are met by operation of the proposed NMSS-MPC control scheme.

Case B

A near non-stationary disturbance $\eta(k)$ is added to the relative humidity of the air, where

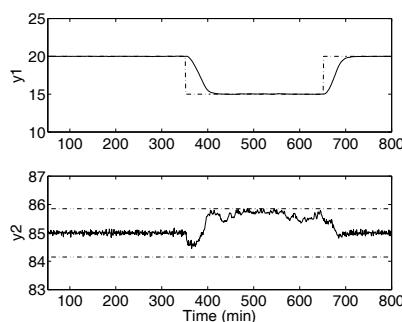
$$\eta(k) = \frac{0.05q^{-1}}{1 - 0.9999q^{-1}}\epsilon(k), \quad (9.37)$$

and $\epsilon(k)$ is a white noise with zero mean and variance of 1. Here the near non-stationary disturbance simulates a drift in the humidity of the air. Figures 9.7a and 9.7b show the control signal responses and constrained rates of change on both control signals, while Figure 9.7c shows the constrained output responses respectively. It is interesting to note that with this near non-stationary disturbance, not only constraints on the rate of change on both control signals are activated, but also the one on the amplitude of control signal u_2 is activated. Nevertheless, all control objectives are met.



(a) Closed-loop control signal response. top figure: valve aperture ($20 \leq u_1(k) \leq 60$); bottom figure: mist spraying ($2 \leq u_2(k) \leq 18$)

(b) Rate of changes in control signal response. Top figure: rate of change for valve aperture ($-1.03 \leq \Delta u_1(k) \leq 1.35$); bottom figure: rate of change for mist spraying ($-1.03 \leq \Delta u_2(k) \leq 1.15$)

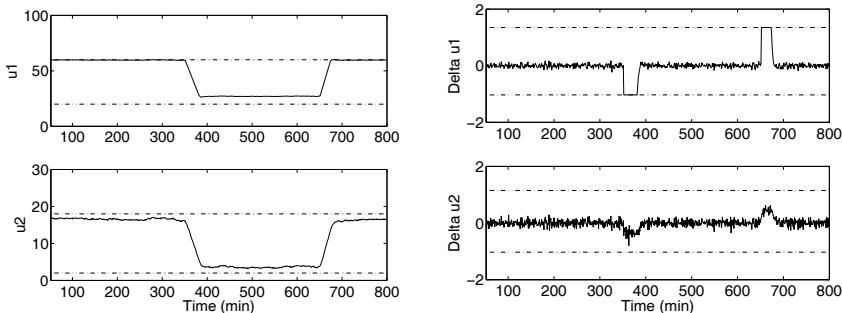


(c) Closed-loop output response. Top figure: air temperature ($15 \leq y_1(k) \leq 20$); bottom figure: relative humidity of the air ($85 - 0.85 \leq y_2(k) \leq 85 + 0.85$)

Fig. 9.7. Case B. Non-stationary disturbance is added

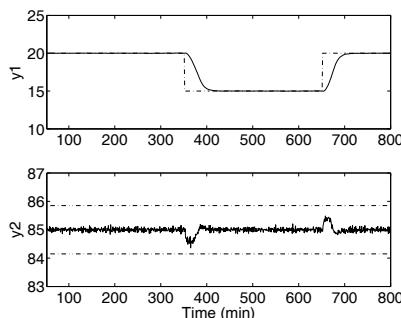
Case C

In the simulation environment of Case B, the robustness of the NMSS -MPC system is examined. The simulated plant is assumed to have an increase of steady state gain of 25%. Figures 9.8a to 9.8c show the closed-loop responses with respect to operational constraints. The results indicate that the design is robust with respect to a small modelling error in the steady-state gain.



(a) Closed-loop control signal response. Top figure: valve aperture ($20 \leq u_1(k) \leq 60$); bottom figure: mist spraying ($2 \leq u_2(k) \leq 18$)

(b) Rate of changes in control signal response. Top figure: rate of change for valve aperture ($-1.03 \leq \Delta u_1(k) \leq 1.35$); bottom figure: rate of change for mist spraying ($-1.03 \leq \Delta u_2(k) \leq 1.15$)



(c) Closed-loop output response. Top figure: air temperature ($15 \leq y_1(k) \leq 20$); bottom figure: relative humidity of the air ($85 - 0.85 \leq y_2(k) \leq 85 + 0.85$)

Fig. 9.8. Case C. Robustness is examined

9.5 Continuous-time NMSS model

Suppose that a continuous-time system is represented by the input and output Laplace transform relation:

$$A(s)Y(s) = B(s)U(s) + \frac{C(s)}{s}\xi(s), \quad (9.38)$$

where

$$\begin{aligned} A(s) &= s^n + a_1s^{n-1} + \dots + a_n \\ B(s) &= b_1s^{n-1} + b_2s^{n-2} + \dots + b_n, \end{aligned}$$

and the integrated noise model $\frac{C(s)}{s}$ with $C(s)$ being a stable polynomial with order m and unit leading coefficient. The Laplace transforms of input, output, and disturbance signals are $U(s)$, $Y(s)$, and $\xi(s)$, respectively.

Since a continuous-time control system involves the derivatives of the plant input and output signal, this operation would amplify the existing noise in the system. For implementation purposes, a stable and all-pole filter is utilized here in conjunction with the noise model $\frac{C(s)}{s}$. Let the implementation filter be chosen as

$$\begin{aligned} F(s) &= \frac{t_n}{s^n + t_1s^{n-1} + t_2s^{n-2} + \dots + t_n} \\ &= \frac{t_n}{T(s)}, \end{aligned} \quad (9.39)$$

where $T(s) = C(s)E(s)$, and degree of $E(s) = n - m$, with $E(s)$ being a stable polynomial. Pre-filtering the input and output signal with $F(s)$, (9.38) can be re-written as

$$A(s)F(s)(sY(s)) = B(s)F(s)(sU(s)) + \frac{t_n\xi(s)}{E(s)}. \quad (9.40)$$

The selection of $F(0) = 1$ maintains the same steady-state values of the filtered input and output signals.

Let us define the filtered output signal as

$$\begin{aligned} y_f^n(t) &= L^{-1}\{F(s)s^nY(s)\} \\ y_f^{n-1}(t) &= L^{-1}\{F(s)s^{n-1}Y(s)\} \\ &\vdots \\ y_f(t) &= L^{-1}\{F(s)Y(s)\}, \end{aligned}$$

and the filtered input signal as

$$\begin{aligned} u_f^n(t) &= L^{-1}\{F(s)s^nU(s)\} \\ u_f^{n-1}(t) &= L^{-1}\{F(s)s^{n-1}U(s)\} \\ &\vdots \\ u_f(t) &= L^{-1}\{F(s)U(s)\}, \end{aligned}$$

and the filtered disturbance as

$$\xi_f(t) = L^{-1}\left\{\frac{t_n \xi(s)}{E(s)}\right\},$$

where $L^{-1}\{\cdot\}$ denotes the inverse Laplace operator. Equation (9.40) can be written in the time domain by the filtered input and output relation:

$$y_f^{n+1}(t) + a_1 y_f^n(t) + \dots + a_n \dot{y}_f(t) = b_1 u_f^n(t) + b_2 u_f^{n-1}(t) + \dots + b_n \dot{u}_f(t) + \xi_f(t). \quad (9.41)$$

The filtered input signal is related to the derivative of the actual control signal $\dot{u}(t)$, through

$$u_f^{n+1}(t) = -t_1 u_f^n(t) - t_2 u_f^{n-1}(t) - \dots - t_n \dot{u}_f(t) + t_n \dot{u}(t), \quad (9.42)$$

while the derivative of the actual output signal $\dot{y}(t)$ is related to the filtered output signal $y_f(t)$ through

$$\begin{aligned} t_n \dot{y}(t) &= (t_1 - a_1) y_f^n(t) + (t_2 - a_2) y_f^{n-1}(t) + \dots + (t_n - a_n) \dot{y}_f(t) \\ &\quad + b_1 u_f^n(t) + b_2 u_f^{n-1}(t) + \dots + b_n \dot{u}_f(t) + \xi_f(t). \end{aligned} \quad (9.43)$$

The predictive control design relies on the use of (9.41), (9.42) and (9.43) to generate the predicted output responses. In the continuous-time domain, we compute prediction of the output by utilizing a non-minimal state-space formulation.

The state vector is chosen as:

$$x(t)^T = [y_f^n(t) \ y_f^{n-1}(t) \ \dots \ \dot{y}_f(t) \ u_f^n(t) \ u_f^{n-1}(t) \ \dots \ \dot{u}_f(t) \ y(t)].$$

Then the model is defined as follows,

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B\dot{u}(t) + E_m \xi_f(t) \\ y(t) &= Cx(t), \end{aligned} \quad (9.44)$$

where

$$A = \begin{bmatrix} -a_1 & -a_2 & \dots & -a_{n-1} & -a_n & b_1 & \dots & b_{n-1} & b_n & 0 \\ 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & 0 \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & -t_1 & \dots & -t_{n-1} & -t_n & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 1 & 0 & 0 \end{bmatrix}; \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ t_n \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \frac{t_1 - a_1}{t_n} & \frac{t_2 - a_2}{t_n} & \frac{t_3 - a_3}{t_n} & \dots & \frac{t_n - a_n}{t_n} & \frac{b_1}{t_n} & \frac{b_2}{t_n} & \dots & \frac{b_n}{t_n} & 0 \end{bmatrix}$$

$$C = [0 \ 0 \ 0 \dots 0 \ 0 \ 0 \dots 0 \ 1]$$

$$E_m = [1 \ 0 \ 0 \dots 0 \ 0 \ 0 \dots 0 \ \frac{1}{t_n}].$$

Note that for a given n th-order transfer function, this particular state-space realization has dimension $2n + 1$ that includes an augmented state variable for integral action.

In the realization above, the pair (A, B) is controllable. The pair (A, C) is not observable from the output, therefore, when the predictive control scheme is chosen to optimize the output response, the unobservable modes in the system are not to be changed by output feedback control. This is achieved in the state feedback framework by choosing $Q = C^T C$. In this case, the poles of the polynomial $F(s)$ form part of the full set of closed-loop poles.

The filtered plant input and output signals can be conveniently realized using the well-known state variable filter framework, where to obtain the derivatives of filtered output responses, a state variable vector X^y is defined as

$$X^y(t) = [y_f^n(t) \ y_f^{n-1}(t) \dots \dot{y}_f(t)]^T.$$

With the state-space model in a control canonical form, we have

$$\begin{bmatrix} \frac{dy_f^n(t)}{dt} \\ \frac{dy_f^{n-1}(t)}{dt} \\ \vdots \\ \frac{dy_f^1(t)}{dt} \end{bmatrix} = \begin{bmatrix} -t_1 & -t_2 & \dots & -t_n \\ 1 & 0 & \dots & 0 \\ \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} y_f^n(t) \\ y_f^{n-1}(t) \\ \vdots \\ y_f(t) \end{bmatrix} + \begin{bmatrix} t_n \\ 0 \\ \vdots \\ 0 \end{bmatrix} \dot{y}(t)$$

$$= A_F \begin{bmatrix} y_f^n(t) \\ y_f^{n-1}(t) \\ \vdots \\ y_f(t) \end{bmatrix} + B_F \dot{y}(t). \quad (9.45)$$

The solution of the state-space equation (9.45), assuming zero initial conditions, gives the derivatives of the filtered output responses. Similarly, define a state vector of filtered control signal

$$X^u(t) = [u_f^n(t) \ u_f^{n-1}(t) \dots u_f^1(t)]^T.$$

These state variables satisfy the following differential equation, assuming zero initial conditions

$$\dot{X}^u(t) = A_F X^u(t) + B_F \dot{u}(t). \quad (9.46)$$

Up to this point, the plant model is assumed to be a single-input, single-output system. For a multi-input, multi-output system, by assuming that the number of inputs equals the number of outputs, a left matrix fraction (LMF) representation of input and output relationship leads to a similar expression of continuous-time state space structure as in (9.44). It is worthwhile to mention that the coefficient matrices t_i , $i = 1, 2, \dots, n$ are chosen to be diagonal matrices in the case of a multi-input and multi-output system.

9.6 Case Studies for Continuous-time MPC

The continuous-time NMSS based MPC is a special case of the general design of continuous-time MPC using Laguerre functions. The central idea is that instead of assuming that the state variables at time t_i are not available, the continuous-time NMSS model provides the realization of the state variables through filtered measurement of plant input and output signals, hence, the use of an observer is avoided. The entire design framework proposed in Chapters 6, 7 and 8 is carried through to the current chapter. Here, we will focus on the special issues raised by the continuous-time state-space structure.

The Role of Filter $F(s)$

The filter $F(s)$ used in the continuous-time MPC plays an important role in determining the performance of the closed-loop system. Because the continuous-time state-space model is not observable, by choosing the cost function as the minimization of the integral squared output error, the poles of $F(s)$ are a subset of the closed-loop poles, hence their locations can be selected in conjunction with the consideration of dynamic response speed, disturbance rejection, and robustness of the closed-loop MPC system. The example considered in this section illustrates the effect of the filter parameters on these important properties of the control system.

Consider an unstable system with the continuous-time transfer function

$$G(s) = \frac{1}{(s^2 - 2s + 2)(s + 5)}, \quad (9.47)$$

and a sampling interval for control of 0.0004 sec. The very small sampling interval is used in the simulation study to attain numerical stability in the environment of discrete simulation. This system has a pair of complex poles at $s = 1 \pm j$. In the design of predictive control, the weight matrices in the cost function are selected as $Q = C^T C$ and $R = 0.01$. The selection of Q and R ensures that the dominant closed-loop poles of the predictive control are close to $s = -1 \pm j$. Based on this observation, the scaling factor in the Laguerre function is selected as $p = 0.8$, and the number of terms N is selected to be 4. The prediction horizon is selected approximately as $10/p = 12.5$, approximately 10 times the dominant time constant of Laguerre functions. By doing so, the Laguerre functions approximate the underlying optimal control trajectory closely. The four closed-loop poles that result from the predictive control design are

$$[-4.9999 - 1.1008 \pm j0.6805 - 0.9328].$$

The first three closed-loop poles are an approximation of the optimal closed-loop poles of a linear quadratic regulator in a high gain situation ($R = 0.01$). Therefore, when constraints are not imposed, there is little scope for further

improvement in the speed of response by employing different design parameters in the predictive control component. Instead, we vary the locations of the filter poles and obtain different types of closed-loop behaviour. In order to compare the performance with different filter settings, four cases are examined: Case A represents the case of slow filter dynamic response where the three poles of the filter are selected to be $-1, -1.2, -1.3$; Case B where the filter dynamic response is made more rapid by increasing the negativity of the three poles to $-5, -5.2, -5.3$; Case C where the negativity of the three poles is increased further to $-10, -10.2, -10.3$; Case D represents the case of fastest filter dynamic response where the poles of the filter are selected as $-50, -50.2, -50.3$. Table 9.1 gives the details of the closed-loop poles for these four cases. It shows that, as pointed out previously, the closed-loop poles consist of predictive control poles and the filter poles. Table 9.2 shows that

Table 9.1. Closed-loop poles

	Closed-loop poles
Case A	$-1, -1.2, -1.3, -4.99, -1.1008 \pm j0.6805, -0.9328$
Case B	$-5, -5.2, -5.3, -4.99, -1.1008 \pm j0.6805, -0.9328$
Case C	$-10, -10.2, -10.3, -4.99, -1.1008 \pm j0.6805, -0.9328$
Case D	$-50, -50.2, -50.3, -4.99, -1.1008 \pm j0.6805, -0.9328$

the feedback controller gains all change with the location of the filter poles, except the gain related to the output signal $y(t)$. The closed-loop responses for cases A, B and C are compared in Figure 9.9 where a unit set-point signal is applied at time $t = 0$, and a constant, unit input disturbance enters the system at $t = 19$ sec. This demonstrates that the nominal set-point response is independent of the filter pole locations, however, the disturbance rejection speed is largely dependent on these locations. As expected, the closed-loop MPC system has a faster disturbance rejection speed when the filter poles are faster. This means that the filter pole locations can be used as very effective closed-loop tuning parameters.

Table 9.2. State feedback controller gains

	State Feedback Control Gain
Case A	69.8698, 262.4908, -490.5861, 3.2913, 19.2099, 50.0790, 7.8113
Case B	13.8153, 62.2304, -34.2297, 0.0373, 0.6646, 4.4434, 7.8113
Case C	9.1177, 43.8322, -8.5769, 0.0049, 0.1605, 1.8781, 7.8113
Case D	5.8700, 30.5201, 6.9898, 0.0000, 0.0062, 0.3215, 7.8113

Since the filter pole locations affect the gain of the closed-loop control system, they directly affect the robustness of the MPC system. In order to

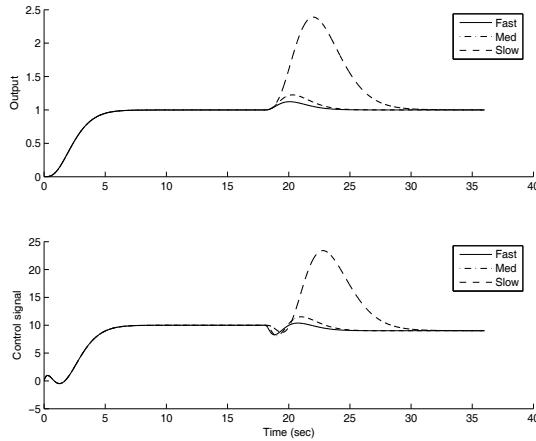


Fig. 9.9. Plant output $y(t)$ (top figure) for Cases A, B and C; and control signal $u(t)$ (bottom figure) for Cases A, B and C

examine this, we reduce the plant gain by 20 percent (*i.e.*, multiply by a factor of 0.8) and simulate the closed-loop response for all the above cases. Figure 9.10 shows the closed-loop response for Cases B, C, and D. The closed-loop system actually becomes unstable for Case A where the filter poles are located at $-1, -1.2, -1.3$, so this is not shown in the figure. As expected, therefore, the closed-loop system has a better tolerance to the model-plant mismatch for the higher feedback gains that are induced by selection of faster responding filter dynamics (see Figure 9.10). This result is consistent with the previous results reported in connection with linear robust control of unstable systems (Wang *et al.*, 1999, Wang and Goodwin, 2000).

The locations of the filter poles play an important role in defining the characteristics of disturbance rejection and noise attenuation in the predictive control system. It has been demonstrated that faster filter dynamics give rise to a higher rate of response to disturbance inputs (see Figure 9.9). However, the fast filter dynamics will tend to amplify the effects of measurement noise in the system. In order to illustrate this, Case C is compared with Case D in simulation studies where integrated white noise is added to the input, while a discrete white noise is added to the output. The control objective is to maintain the output signal at the steady state value (unity). Figure 9.11 compares the output and control signal responses in the presence of these stochastic disturbances. Again, as expected, this demonstrates that although the disturbance rejection response is slower when the filter poles are at the vicinity of $s = -10$, the control signal is much less affected by the measurement noise than when they are selected at around $s = -50$ and so is more acceptable in practical terms.

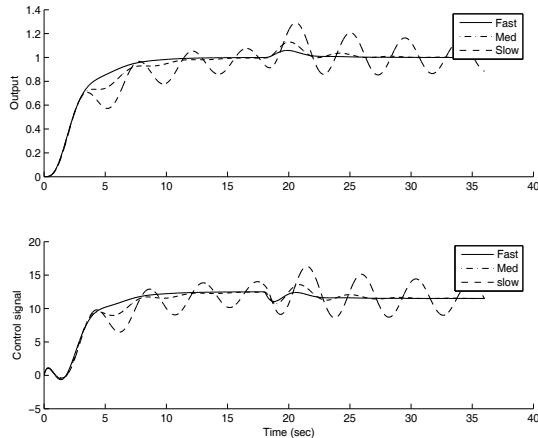


Fig. 9.10. Plant output $y(t)$ (top figure) for Cases B,C and D; and control signal $u(t)$ (bottom figure) for Cases B, C and D

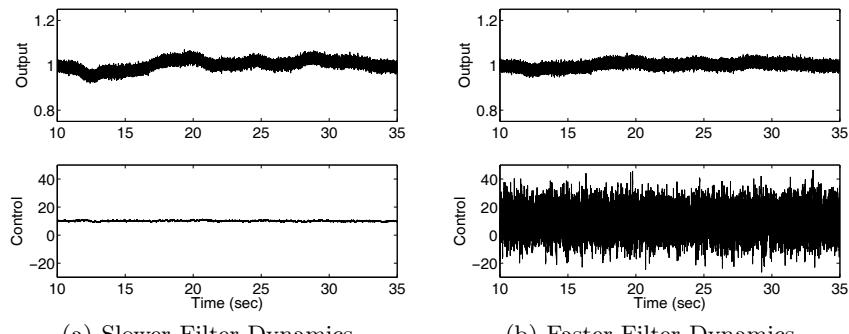


Fig. 9.11. Effect of filter dynamics on disturbance rejection and noise attenuation

9.7 Predictive Control Using Impulse Response Models

The key feature of the dynamic matrix control (DMC) algorithm, different from GPC, is the use of non-parametric models in its design. Non-parametric models are typically step response models or impulse response models, both having been widely used in the process industries. Both algorithms solve for Δu by minimizing the predicted errors between the set-point signal and the output signal. Plant operational constraints are included as inequality constraints in the numerical solutions.

There are a few reasons from a practitioner's viewpoint why a step response model or an impulse response model is desirable. A unit step response model reveals a plant time constant, time-delay and gain in a transparent way. For example, these parameters can be determined directly via a graphical display.

For this reason alone, step response tests are widely used by process engineers as the first step to access plant information. However, if the disturbances in the plant contain significant low frequency components, and the plant has many inputs and outputs and a long response time, the test by itself often does not yield a sufficiently accurate plant model for predictive control design purposes. Under these circumstances, a more rigorous dynamic test is carried out to estimate a mathematical model for predictive control design using tools from system identification. It is often the case that the finite impulse response (FIR) models are directly estimated from plant test data, which are then easily converted to step response models. The plant output is typically expressed via an FIR model as

$$y(k) = h_1 u(k-1) + h_2 u(k-2) + \dots + h_n u(k-n) + \frac{\epsilon(k)}{1 - q^{-1}}, \quad (9.48)$$

where the model coefficients, h_1, h_2, \dots, h_n form the response of a unit impulse input with sampling instant at $k = 0, 1, \dots, n$. If there are d samples of time delay, then the first d coefficients will be zero. Parameter n is the number of the parameters included in the model, which is often estimated using the rule $n \approx \frac{T_s}{\Delta t}$, where T_s is the plant settling time and Δt is the sampling interval. For instance, if T_s is about 100 sec, and the sampling interval is 1 sec, then the estimated n is 100, meaning that there will be 100 parameters required to capture the dynamic response via an FIR model or equivalently, a step response model.

Because the FIR models contain input-only variables (right hand side of (9.48)), which will form a linear regressor that has only input variables, a simple least squares algorithm will produce consistent estimation results from open-loop test data, under mild statistical conditions (see Ljung, 1999). In addition, the FIR model structure requires less *a priori* information than the general transfer function models. For instance, a general transfer function model requires the information of time delay and model order. Furthermore, a mismatch of a general transfer function model structure could cause bias in the estimated parameters. In contrast, an FIR model only needs the information of settling time which is often available or easy to identify. These advantages of FIR become even more relevant for a complex situation when the plant has many input and output variables and has complicated dynamic responses due to interactions and inner-loop feedback and feedforward.

For these reasons and arguments, FIR types of models have been used in classical dynamic matrix control, and gained acceptance in the process industry for the past three decades.

When a FIR model is obtained, the corresponding non-minimal state-space model is expressed as a special case in Section 9.3, where the denominator takes the form $F(z) = I$, I is the identity matrix. However, the number of terms for the $H(z)$ is increased to cover the dynamic response of the plant. Thus all the $\Delta y(\cdot)$ terms vanish from the state variable vector. Instead, the state vector takes the form:

$$x(k) = [\Delta u(k-1)^T \ \Delta u(k-2)^T \ \dots \ \Delta u(k-n)^T \ y(k)^T].$$

The plant information at sampling time k_i is captured by the current measurement $y(k_i)$ and the past $\Delta u(k-1), \Delta u(k-2), \dots, \Delta u(k-n)$.

There are a few remarks to be made at this point. Although there are many advantages in using a FIR model in the step of model identification, the advantages apparently are lost when using the estimated FIR model directly for predictive control design. It is seen that in order to capture the plant information at time k_i , the state variable vector $x(k_i)$ requires all the past incremental control signals up to the model order n . The dimensionality of $x(k_i)$ could be very large if the system is in a fast sampling environment (Δt is small). The consequence of a large model dimension is the increase of computational load in real time. Another disadvantage of the direct use of a FIR model is that although the bias of the estimated parameter vector is small, the variances could be larger than those from a model identified in a compact transfer function form. Larger variances are typically reflected in a non-smooth behaviour in the impulse response constructed from the model coefficients. These random errors in a high dimensional matrix could be amplified in the computation of predictive control gain matrices.

In order to reach a compromise between the FIR model and a compact transfer function model with denominator and numerator, we need to use a more general class of input-only variable models. The Laguerre models are selected for this purpose. This class of models can be used to replace a FIR model with an appropriate scaling factor to improve its efficiency. We briefly mentioned this previously in Chapter 3. In this regard, suppose that the impulse response of a stable system is $h(k)$, then with a given number of terms N , $h(k)$ is written as (see Chapter 3)

$$h(k) = c_1 l_1(k) + c_2 l_2(k) + \dots + c_N l_N(k). \quad (9.49)$$

Choosing the state variable vector $x_m(k)$ as the filtered input $u(k)$ by the Laguerre networks, with an arbitrary input signal $u(k)$, the output signal $y(k)$ is

$$x_m(k+1) = A_l x_m(k) + B_l u(k) \quad (9.50)$$

$$y(k) = C_l x_m(k), \quad (9.51)$$

where

$$A_l = \begin{bmatrix} a_p & 0 & 0 & 0 & 0 \\ \beta_p & a_p & 0 & 0 & 0 \\ -a_p \beta_p & \beta_p & a_p & 0 & 0 \\ a_p^2 \beta_p & -a_p \beta_p & \beta_p & a_p & 0 \\ -a_p^3 \beta_p & a_p^2 \beta_p & -a_p \beta_p & \beta_p & a_p \end{bmatrix}$$

$$B_l = \sqrt{(1 - a_p^2)} [1 \ -a_p \ a_p^2 \ -a_p^3 \ a_p^4]$$

$$C_l = [c_1 \ c_2 \ c_3 \ c_4 \ c_5].$$

Parameter a_p is the scaling factor, $\beta_p = 1 - a_p^2$ and $N = 5$ for this case. Note that the subscript p is used for the scaling factor so as to differentiate this dynamic model from the general design framework of MPC using Laguerre functions.

By using similar manipulation as before, the augmented state-space model is expressed as

$$x(k+1) = Ax(k) + B\Delta u(k), \quad (9.52)$$

where

$$x(k) = \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}; A = \begin{bmatrix} A_l & o_m^T \\ C_l A_l & 1 \end{bmatrix}; B = \begin{bmatrix} B_l \\ C_l B_l \end{bmatrix}; C = [o_m \ 1],$$

o_m^T is a zero column vector with its dimension equal to that of $x_m(k)$.

Note that in this formulation, the state variable vector $x(k)$ is available from the measurement of the output $y(k)$ and the optimal incremental control $\Delta u(k)$. Thus, there is no need for an observer in the implementation. More precisely, the following equation is used to generate the feedback variable $\Delta x(k)$:

$$\Delta x_m(k+1) = A_l \Delta x_m(k) + B_l \Delta u(k), \quad (9.53)$$

where the initial condition of $\Delta x_m(k)$ at $k = 0$ is assumed to be a zero vector. The plant information at the beginning of the optimization window (sample time k_i) is captured by the state variable $\Delta x_m(k_i)$ (see (9.53)) and the current measurement of the output signal $y(k_i)$. The design framework in discrete-time MPC is carried over to the special case where the state variables are selected as filtered responses of the $\Delta u(k)$ and the output $y(k)$. This realization is directly associated with the advantages of using this type of dynamic models in system identification.

The discrete Laguerre model structure has been used in adaptive control (Zervos and Dumont, 1988) and predictive control (see Finn *et. al.*, 1993). Those who are interested in finding plant step response models can follow the two-step approach in Wang *et al.* (2003) where the first step was to compress the plant test data to step responses of sub-systems, followed by a second step of fitting each step response with a Laguerre model which will smooth the response, reduce variances and the dimension of the subsystem model. The second step was discussed in Wang and Cluett (2000).

9.8 Summary

This chapter has re-visited classical model predictive control using a state-space formulation. The classical model predictive control systems such as GPC and DMC directly utilize plant input and output information in their feedback structures and avoid the use of an observer in the implementation.

By using non-minimal state-space realizations in both continuous-time and discrete-time domains, these classical model predictive control systems are re-formulated in terms of a state-space model. Because of the state-space formulation, all the previous design and analysis in both continuous time and discrete time can be carried through to the classical predictive control systems, including guaranteed asymptotic stability and numerically a well-conditioned solution with exponential weighting.

In addition, these non-minimal state-space realizations lead to simplified design procedures that do not require the design and implementation of observers. In discrete time, it is not necessary to filter the input and output variables, as demonstrated in the chapter. However, if there is severe noise present in the system, a noise filter can be implemented to reduce the effect of high frequency noise, by following the same procedure as the introduction of a continuous-time filter in the continuous-time formulations presented. In continuous time, it is necessary to use an implementation filter to avoid differentiation of the input and output measurement variables. The filter poles become part of the closed-loop poles by choosing the weight matrix $Q = C^T C$ in the cost function in predictive control. This is convenient because the filter design problem is solved in a much easier manner than the one where an observer is used. It is worthwhile to mention that for a discrete-time single-input and single-output system, using the NMSS realization, the implementation of a predictive control system with constraints is in fact easy. The tuning of the predictive control system is also straightforward.

The initial idea of using the NMSS model in the design of a discrete-time MPC system was published in Wang and Young (2006) and the continuous-time counterpart was published in Wang *et al.*, (2009).

Problems

9.1. A dynamic system model is often estimated from plant input and output experimental data. Assume that a discrete-time transfer function is estimated from a set of experimental data with the following form:

$$G(z) = \frac{z - 0.1}{(z - 0.8 + j0.1)(z - 0.8 - j0.1)} z^{-3}.$$

The system is known to have a disturbance in the nature of random walk, which is described by

$$d(k) = \frac{0.005\epsilon(k)}{1 - 0.999z^{-1}},$$

where $\epsilon(k)$ is white noise with zero mean and unity variance.

1. Choose the state vector $x_m(k) = [y(k) \ y(k-1) \ u(k-1) \ u(k-2) \ u(k-3) \ u(k-4)]^T$, and obtain the discrete-time NMSS model. Augment this NMSS model with an integrator.

2. Design a predictive control system that contains an integrator. The weight matrices in the cost function are chosen as: $Q = C^T C$ where C is the output matrix of the augmented state-space model, $R = 1$. The prediction horizon $N_p = 48$ and the exponential weight factor $\alpha = 1.2$ to produce a numerically sound solution. The parameters in the Laguerre functions are $a = 0.6$ and $N = 6$.
3. Simulate closed-loop performance of this predictive control system with $d(k)$ added to the input signal.

9.2. Continue from Problem 9.1. Suppose that the plant output is given a set-point change at the simulation time $k = 20$, with the input disturbance assumed present at all simulation time. The system operational constraints are

$$-0.2 \leq \Delta u(k) \leq 0.2; 0 \leq u(k) \leq 0.25; -0.2 \leq y(k) \leq 1.2.$$

Impose the constraints and demonstrate a model predictive control designed using the discrete-time NMSS model structure.

9.3. Frequency response analysis of a feedback control system provides valuable information about the robustness of the closed-loop system. When using the NMSS structure, because the feedback variables are directly linked to the plant input and output signals, the predictive controller, without imposing constraints, can be expressed in a polynomial form: $u(z) = -C(z)y(z)$. A Nyquist plot is the graphical presentation of the open-loop frequency response of a control system (Goodwin *et al.* 2000). Here the open-loop frequency response is $G(e^{jw})C(e^{jw})$, where $0 \leq w \leq \pi$. Generate a Nyquist plot for the predictive control system designed in Problem 9.1. What are the gain and phase margins for this predictive control system? Validate your answers using closed-loop simulation.

9.4. A continuous-time NMSS model is used for the design of a continuous-time predictive control in the state-space formulation. A continuous-time system is described by a transfer function model:

$$G(s) = \frac{2}{(10s + 1)^3}. \quad (9.54)$$

Design and implement a continuous-time predictive control system that will follow a step set-point signal and reject a step input disturbance. The weight matrices in the cost function are chosen as $Q = C^T C$, where C is the output matrix of the augmented model, $R = 1$. The parameters for the Laguerre functions are $p = 1$ and $N = 3$. The prediction horizon $T_p = 20$ and the exponential weight factor $\alpha = 0.5$. The filter poles are tuning parameters, which are often set according to the response speed of disturbance rejection. Choose the filter poles as -0.1 for slow response; -1 as medium response speed; and -5 as fast response speed. Simulate the closed-loop performance of the predictive control system with a unit set-point change at $t = 0$ and a

unit input step disturbance at $t = 20$. The sampling interval is $\Delta t = 0.001$ for the simulation.

9.5. A continuous-time predictive control system has advantages over its discrete-time counterpart when the system operates in a fast sampling environment. A continuous-time predictive control system is designed using a continuous-time model, and the discretization occurs at the final implementation stage. In contrast, a discrete-time system is designed using a discrete-time model so the discretization occurs at the beginning of the design. Discretize the continuous-time transfer function model (9.54) given in Problem 9.4 using sampling interval $\Delta t = 0.001$ to obtain the corresponding z -transfer function $G(z)$. With such a fast sampling rate, what happens to the parameters in $G(z)$? Design a discrete-time NMSS model predictive control system based on $G(z)$. Choose all the performance parameters as you desire with the aim to match the closed-loop performance of a continuous-time NMSS predictive control system. What are the difficulties for design and implementation of a discrete-time predictive control system in a fast sampling environment?

9.6. Step response or impulse response models are flexible in nature. They are often used in the process industry for describing complex dynamic systems. Consider the process given by the transfer function (Wang and Cluett, 2000)

$$G(s) = \frac{K_p R (\tau_2 s + 1) e^{-d_1 s}}{(\tau_1 s + 1)(\tau_2 s + 1 - (1 - R)e^{-d_2 s})}, \quad (9.55)$$

where $d_1 = 45$, $d_2 = 75$, $K_p = 0.8$, $R = 0.6$, $\tau_1 = 5$ and $\tau_2 = 10$. This transfer function was obtained for a process that has a recycle stream. Choosing $\Delta t = 3$, obtain the set of impulse response coefficients for this system (SIMULINK simulation can be used to obtain the impulse response). Design a discrete-time NMSS predictive control system for this recycle plant. The weight matrices in the cost function are $Q = C^T C$, and $R = 1$, where C is the output matrix of the augmented model. The parameters in the Laguerre functions are $a = 0.2$ and $N = 6$. The prediction horizon is $N_p = 100$ with exponential weight factor $\alpha = 1.2$. Assuming a unit step set-point change at $k = 0$ and a unit step input disturbance at $k = 100$, simulate the nominal closed-loop response of this predictive control system. Determine whether the closed-loop system will be stable when it is implemented on the original plant by examining the Nyquist plot of the predictive controller and the original plant model (the recycle system).

Implementation of Predictive Control Systems

10.1 Introduction

This chapter presents three different implementation procedures for model predictive control systems. The first implementation is based on a micro-controller (low cost) for controlling a DC motor. In this application, the MATLAB® design programs, ‘mpc.m’ and associated functions, are utilized to calculate the predictive controller gain and the previous MATLAB closed-loop simulation program is converted to a C program for real-time implementation on the micro-controller. The procedure is straightforward for those who understand C language. The second implementation is based on MATLAB Real-time Workshop and xPC target. This application is very useful for those who are not familiar with C language because the MATLAB Real-time Workshop and xPC target perform the conversion from MATLAB programs to C programs through the compilers in a systematic way. With these tools, we only need to create MATLAB embedded functions for the real-time applications. The third implementation uses the platform of a real-time PC-based supervisory control and data acquisition (SCADA) system. A pilot food extrusion plant is controlled by the continuous-time predictive controller developed in Chapter 6. In this application, the MATLAB program ‘cmpc.m’ is used to generate the predictive controller gain and the previous MATLAB closed-loop simulation program for a continuous-time system is converted to a C program for the real-time implementation.

10.2 Predictive Control of DC Motor Using a Micro-controller

In this application, a pair of low-cost brushed DC motors are used to control the elevation and azimuth angles of a solar concentrator used for electrical power generation. The concentrator requires very precise positional control to

enable the Sun's energy to be constantly and accurately focused onto a photo voltaic array. Most of the concentrator's movements are slow (nominally 15 degrees per hour) and predictable, however under emergency conditions the concentrator will be required to move relatively rapidly (full motor RPM).¹

10.2.1 Hardware Configuration

The core controller is a low-cost microchip single chip digital signal processor (DSP). The chosen part is DSPIC30F6014 that contains a 16 bit processor, and has multiple on-chip pulse width modulations (PWMs). Figure 10.1 shows a photograph of the 2-axis controller. Note that this unit also provides data-logging functions, and therefore has other unrelated hardware. Apart from the Microchip or DSP, the plant in the study is a DC motor. A photograph of the combined motor, encoder and gear head is shown in Figure 10.2. Note the encoder, which consists of a pair of hall effect sensors, and a multi-pole circular magnet.



Fig. 10.1. Micro-controller

¹ Mr Stephen Gale, in Elektrika and RMIT, designed and implemented the hardware and software for this low-cost MPC application. The total cost of the core micro-control system was about 12 US dollars.

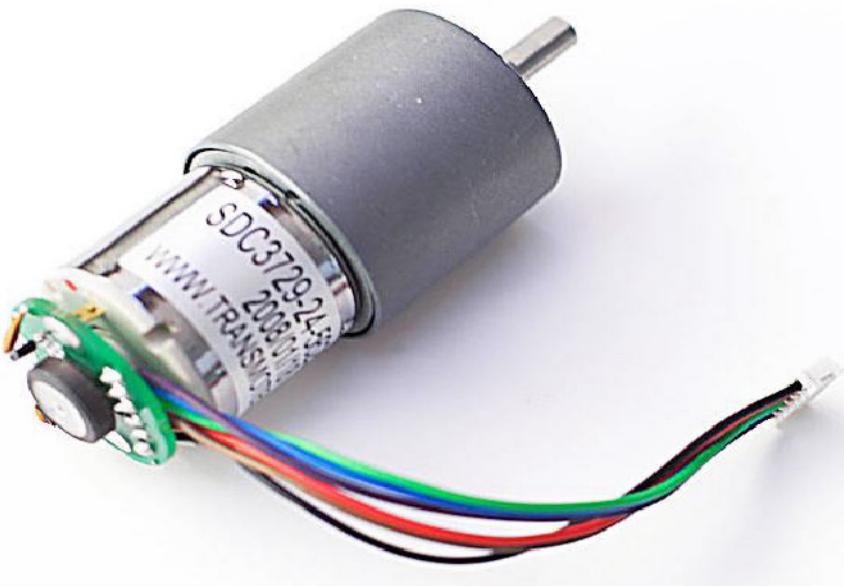


Fig. 10.2. DC motor

Functionalities of the Micro-controller

In the configuration, the following tasks are performed by the micro-controller.

1. Compute the required concentrator position in terms of encoder counts.
2. Read the 2-bit discrete output of a pair of digital shaft encoders, and convert the outputs to shaft position.
3. Compute the latest motor commands using the discrete-time model predictive algorithm described in Chapter 3.
4. Convert the motor command to a set of variable pulse width signals.
5. Amplify the PWMs output to provide a bipolar 0 to 24 V signal at up to 5 A.
6. Provide a serial communications link to enable the effectiveness of the control scheme to be evaluated.

Some of the functions are performed in software. In particular, this DSP does not have an in-built counter to convert the two phase encoder signal from the encoder into a count value. Instead, the DSP was programmed to jump to an interrupt routine on a change of state of either of the encoder outputs. A look-up table was then used to determine if the software encoder counter needs to be incremented or decremented. The DSP has on-chip PWMs which are used to drive the lower halves of each of the H bridges. A software routine determines which arm of the bridge is to be driven, and which high-side driver

is required to be enabled, depending on the intended direction of travel. A current shunt in series with the low side driver pairs is coupled to an amplifier and applied to the DSPs on-chip analogue to digital converter (ADC) so that current feedback can be employed if required. This was not used in this series of experiments.

Finally, the DSPs serial port was configured to output real-time measurement data. A National Instruments LabView application was created to accept this data and create the performance plots shown later.

10.2.2 Model Development

Several model structures are available to capture the motor's electrical equivalent circuit. The simplest model has a first-order plus integrator structure:

$$G_m(s) = \frac{K_p}{s(\tau_p s + 1)}, \quad (10.1)$$

where the output is angular position, K_p is the gain and τ_p is the time constant related to the mechanical part of the motor. A sample interval of $\Delta t = 20$ ms was selected as a starting point. Here, the sample interval was chosen as a compromise between the position encoder accuracy response and the controller response time. In other words, for an excessively short sampling interval, if the motor is turning slowly, the encoder will have changed very little or not at all, resulting in undesirably large steps in the calculated motor command. Other sampling intervals of 10 ms and 50 ms were later tested, however, the original choice provided the smoothest response.

To estimate the gain term K_p and the time constant τ_p , an experiment was set up to provide the motor with a square wave control input, oscillating between 25 percent and 75 percent of full scale. Note for this PWM and software and hardware configuration, a control input of 100 percent corresponds to an input value of 4000 counts. In addition, to reverse the motor and drive it at 100 percent speed, a value of -4000 is used. A separate routine selects the required PWM and selects the correct control signal that corresponds to the desired high-side driver.

Figure 10.3 shows the angular velocity as seen by the encoder with the 25 to 75 percent PWM command. From this, we estimate the gain term to be approximately $K_p = 0.0125/0.02$. Because the number of samples to steady state is about 30 for increasing the speed and 50 samples for decelerating, the response time is about $30 \times 20 = 600$ ms for acceleration and $50 \times 20 = 1000$ ms for deceleration. This motor accelerates faster than it decelerates. Since in our application of position control we are more interested in the motor's ability to stop in the correct position, the time constant was determined from the decelerating side of the curve. Thus, the time constant $\tau_p = \frac{1000}{5} = 200$ ms = 0.2 s, which is about one-fifth of the step response time.

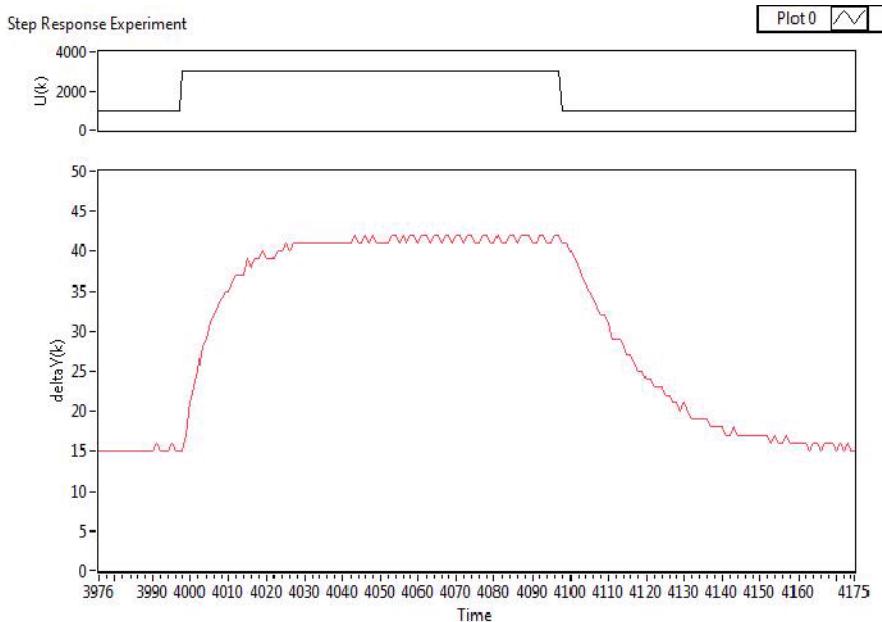


Fig. 10.3. Step response test of motor dynamics

10.2.3 DMPC Tuning

The discrete-time model predictive control system uses the non-minimal state-space model as in Chapter 9 where the plant input and output are selected as state-variables, hence avoiding use of an observer. The design has also been discussed in Section 3.8 to demonstrate the closed-form solution for single-input and single-output system.

The z -transfer function for the motor is obtained using a sampling interval of 0.02 (sec) with zero-order hold, as

$$G(z) = \frac{(0.6047z + 0.5849) \times 10^{-3}}{z^2 - 1.9048z + 0.9048}, \quad (10.2)$$

where MATLAB function c2dm is used.

The state variable $x(k)$ is chosen to contain the plant input and output variables as

$$x(k) = [\Delta y(k) \ \Delta y(k-1) \ \Delta u(k-1) \ y(k)]^T,$$

then, following from Chapter 9, the augmented NMSS model is defined as

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k), \end{aligned} \quad (10.3)$$

where the matrices A, B, C are

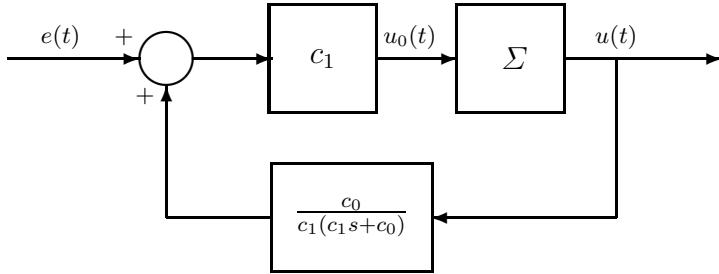


Fig. 10.4. PI controller with anti-windup mechanism

$$A = \begin{bmatrix} 1.9048 & -0.9048 & 0.0006 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1.9048 & -0.9048 & 0.0006 & 1 \end{bmatrix}; B = \begin{bmatrix} 0.0006 \\ 0 \\ 1 \\ 0.0006 \end{bmatrix}$$

$$C = [0 \ 0 \ 0 \ 1].$$

The closed-loop response speed is not critical, however, overshoot in output response is not desirable. Therefore, in the design of MPC, the parameters in the Laguerre functions are selected to reflect this requirement with $a = 0.95$ and $N = 1$, which restricts the incremental of the control signal to behave like a first order system response. A prediction horizon $N_p = 30$ is used in the design. The weight matrices in the cost function are chosen as $Q = C^T C$ and $R_L = 0.3$, leading to the cost function:

$$J = \sum_{m=1}^{N_p} x(k_i + m | k_i)^T Q x(k_i + m | k_i) + \eta^T R_L \eta. \quad (10.4)$$

Here, R_L is used as part of the performance tuning parameters.

10.2.4 DMPC Implementation

The programs for implementation are written in C language to suit the Microcontroller. Although MATLAB has a compiler that can convert MATLAB programs into C language, in this application, we directly write the real-time executable programs using C Language.

For a SISO system, the majority of the controllers used in industries are PID controllers. The main reason behind the success of PID controllers is the simplicity of the design and implementation. It is useful that we make a comparison between the MPC with NMSS structure and PID controllers.

PID controllers are implemented with an anti-windup mechanism to deal with hard constraints on control signal amplitude and rate of change. There

are numerous books that have discussed the anti-windup schemes (for example, Astrom and Hagglund, 1988, Goodwin *et al.* 2000, Johnson and Moradi, 2005). There are many forms of anti-windup schemes. For instance, Figure 10.4 shows a PI controller with an anti-windup mechanism with amplitude constraint, where Σ represents the saturation nonlinearity. If $u^{min} < u_0(t) < u^{max}$, then $u(t) = u_0(t)$; if $u_0(t) \leq u^{min}$, then $u(t) = u^{min}$; if $u_0(t) \geq u^{max}$, then $u(t) = u^{max}$. If we replace the saturation by a unity gain, then the transfer function from the error signal e to the control signal u is

$$\frac{U(s)}{E(s)} = \frac{c_1 s + c_0}{s}. \quad (10.5)$$

The integral action in this configuration is achieved by putting positive feedback around a stable transfer function. If the control signal reaches a limit, then the integral action will be stopped and the anti-windup scheme will be achieved.

In comparison with predictive control, when the control signal reaches a limit, the integral action in the prediction control is stopped by the solution of optimal $\Delta u(k) = 0$. In addition, this saturation information $\Delta u(k) = 0$ is fed into the state vector $x(k)$ for calculation of the prediction in the next step, as

$$x(k) = [\Delta y(k) \ \Delta y(k-1) \ \Delta u(k-1) \ y(k)]^T.$$

The implementation of a discrete-time MPC control system using a non-minimal state-space model is seen to be relatively straightforward. The task is simpler than a PID controller with saturation, because the saturation in DMPC is implemented as ‘clipping’ the original control signal in the single-input and single-output case when the constraints are violated.

Furthermore, the framework of DMPC using a NMSS structure permits control of higher-order systems with complex dynamics, while the amount of implementation effort remains the same. However, the implementation of a control system on a plant requires many practical skills and a full understanding of the underlying system, particularly if the implementation is carried out with C programs.

10.2.5 Experimental Results

Benefits of Imposing Constraints

Firstly, the control algorithm was tested without constraints. We found that when giving large changes of command position, the predictive control system became unstable. Investigation reveals that when a large change of command position led to a large control signal, without constraints, the control signal would reach its maximum value and would reverse its sign. As a consequence, the closed-loop control system became unstable. For this reason alone the constraint on the magnitude of the control signal $u(k)$ is necessary in the

implementation of predictive control using micro-controllers with limited accuracy. Other benefits of limiting $u(k)$ are related to the integrator effect of the MPC algorithm. Without the constraints on the control amplitude, when $u(k)$ exceeds the limits corresponding to ± 100 percent, or ± 4000 counts in this application, the response to large excursions will be slow while the integrator unwinds. The constraints on Δu provide the benefit of limiting the peak current consumption of the motor. Apart from saving power, this will prolong the life of the motor's brushes and prevent undue stress from being placed on the gear head.

Constrained Control Experiments

The constraints on the control signal are specified as

$$-4000 \leq u(k) \leq 4000,$$

and the constraints on the incremental control Δu are specified as

$$-200 \leq \Delta u(k) \leq 200.$$

In addition, there is a dead-band setup in the implementation to save battery power and avoid unnecessary operation of the control system due to noise in the system. When the error $|y(k) - r(k)| < 12$, the motor will be switched off to avoid wasting battery power; and when the error $|y(k) - r(k)| > 24$, the motor will be switched on to bring the error back within the specified error band.

Figure 10.5 shows the error $y(k) - r(k)$ and control signal $u(k)$ in response to a step input. In this experiment, both constraints on the control signal and the increment of the control signal are not active, so the closed-loop response is the response of a linear system. Figure 10.6 shows the closed-loop system response when both constraints on the amplitude of the control $u(k)$ and increment of the control $\Delta u(k)$ become active. It is seen from this figure that when constraints are active, the predictive control system not only satisfies the operational constraints, but also produces a satisfactory closed-loop response.

10.3 Implementation of Predictive Control Using xPC Target

10.3.1 Overview

Real-time Windows Target provides a PC solution for prototyping and testing of real-time systems. Here, we use this platform for the implementation of predictive control systems. In this application, we use a computer as a host computer and then another computer as the target computer. Figure 10.7 shows

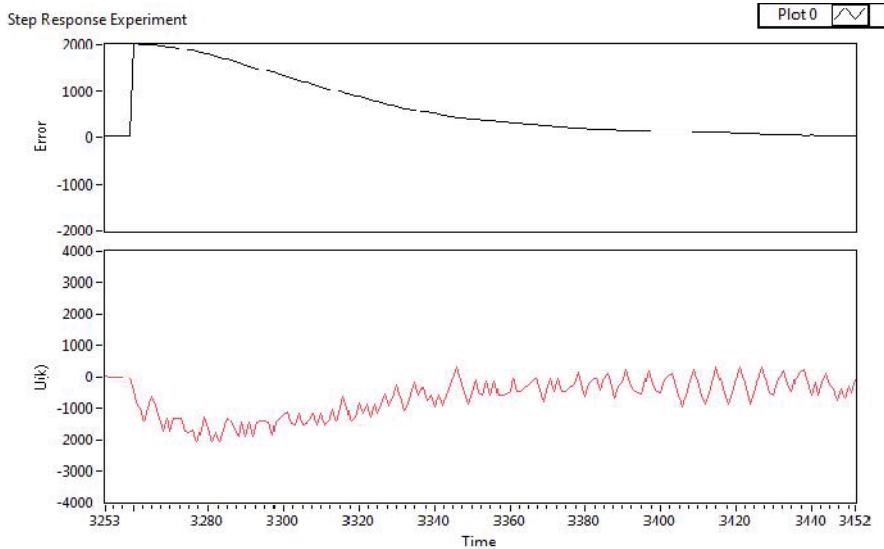


Fig. 10.5. Closed-loop response when constraints are not active. Top plot: the error signal $y(k) - r(k)$; bottom plot: control signal

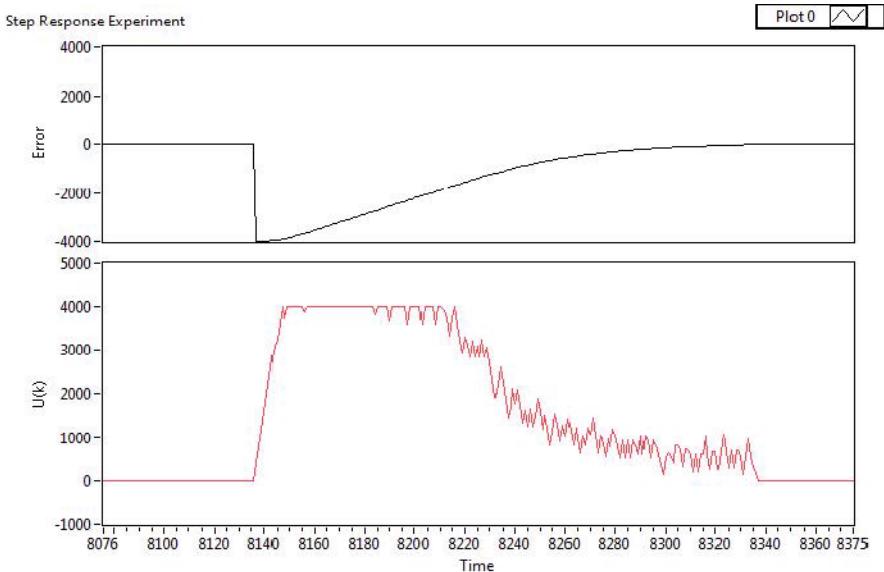
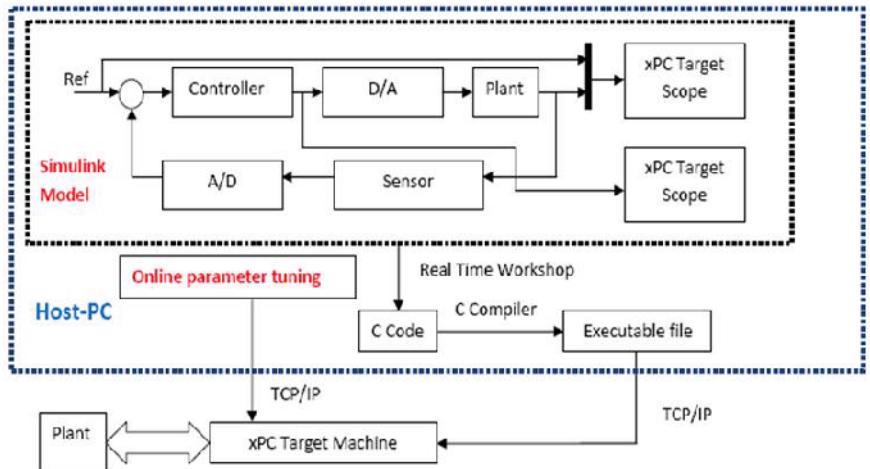


Fig. 10.6. Closed-loop response when constraints are active. Top plot: the error signal $y(k) - r(k)$; bottom plot: control signal

**Fig. 10.7.** xPC block diagram

xPC block diagram. The MATLAB programs we created in the previous chapters will be used in the design of MPC systems, and the simulation programs in MATLAB will be converted into SIMULINK embedded functions. The Real-time Workshop provides utilities to convert the SIMULINK embedded models in C code and then, with the open source Watcom C/C++ compiler, compile the code into a real-time executable file. Although the underlying code is compiled into a real-time executable file via the C/C++ compiler, this conversion is performed automatically without much input from the user. Thus the implementation experience is an extension from previous MATLAB programming experience. This is particularly useful to those who have limited skills in hardware design, implementation and C programming. Another interesting feature of the Real-time Windows Target is that the process from the simulation to implementation is shortened, as illustrated in Figure 10.7. The concept in Figure 10.7 shows that a simulation model can be used in the simulation testing of the predictive control system, and after completing the test, then with simple modification to the original SIMULINK programs, the same real-time predictive control system can be connected to the actual plant for controlling the plant.

10.3.2 Creating a SIMULINK Embedded Function

The key process in using Real-time Workshop is to create SIMULINK embedded functions. The simulation programs we wrote in MATLAB will be converted to SIMULINK embedded functions, in a straightforward manner, except that the dimensions of the variables may be required to be specified explicitly because of the conversion process to C codes.

Tutorial 10.1. The objective of this tutorial is to demonstrate how to create a SIMULINK embedded function from a MATLAB function. We use an observer equation as the test process, where we compute

$$\hat{x}(k+1) = A\hat{x}(k) + B\Delta u(k) + K_{ob}(y(k) - C\hat{x}(k)),$$

where the input variables are $\Delta u(k)$, $y(k)$, and output is $\hat{x}(k+1)$; A , B , K_{ob} are the parameters.

Step by Step

1. Open a new SIMULINK model and save it as ‘observer.mdl’. First, we will choose the configuration parameters so that the file observer.mdl can perform a real-time control task through xPC target.
2. Go to ‘Simulation’ and click to open ‘Configuration Parameters’. Under ‘Solver option’, choose fixed step and discrete; under ‘Periodic sampling constraints’, choose ‘Unconstrained’; Under ‘Fixed step size’, enter the sampling interval value (e.g. 0.02); under ‘task mode for periodic sample time’, choose ‘auto’. Click ‘Apply’ to save and exit.
3. On the left hand side window, find ‘Real-time Workshop’. Click it to open the window. Under ‘System target file’, choose ‘xpctarget.tlc’; Under ‘Language’, choose ‘C’; Click ‘Apply’ to save and exit.
4. We are ready to create an embedded function from a MATLAB function.
5. Click ‘SIMULINK Library’ and find ‘User defined functions’; Drag ‘Embedded MATLAB functions’ into the SIMULINK file ‘observer.mdl’.
6. We will next create the real-time simulation program based on the observer equation. The basic procedure is to type the original MATLAB program into the SIMULINK block first; then define the input and output variables and the parameters.
7. We click to open MATLAB embedded function leading to embedded MATLAB editor; enter the following program into the file for the embedded function:

```
function state_observer = observer(A,B,C,K_ob,y,u_delta)
persistent X_hat
if isempty(X_hat)
X_hat = zeros(6,1);
% X_hat is stored locally in this embedded function,
% we need to define its initial condition.
end
X_hat=A*X_hat+K_ob*(y-C*X_hat)+B*u_delta;
state_observer = X_hat;
```

8. We need to let SIMULINK know, among the input and output variables, which are the parameters, input variables and output variables.

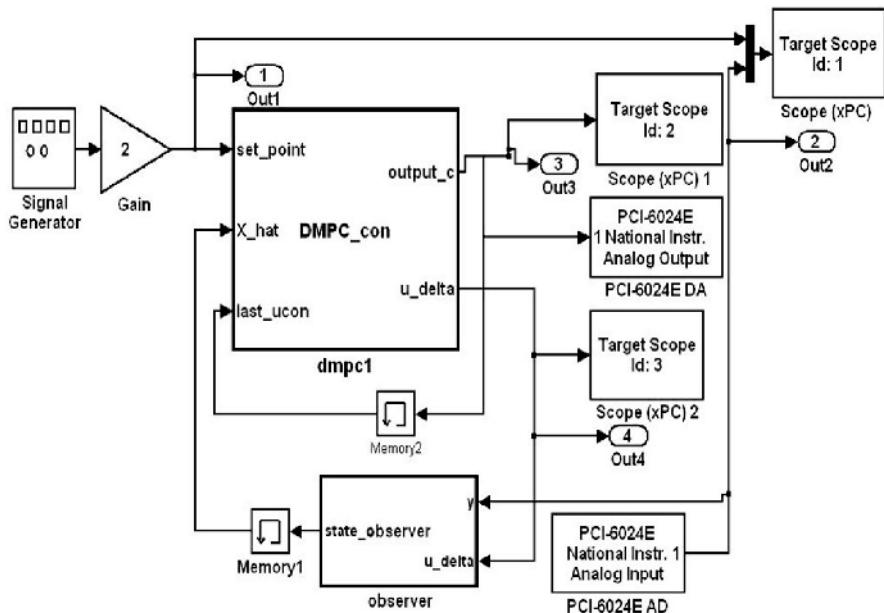


Fig. 10.8. Discrete-time MPC using xPC target

9. Within the embedded MATLAB editor, find ‘Tools’ and click to open ‘Tools’ and select ‘Edit data/ports’ leading to window of ‘data/ports manager’. Change A , B , C , and K_{ob} from the default input variables to parameters, and the input variables remain as y and u_{delta} , the output variable remains as $state_o$.

Figure 10.8 shows the SIMULINK embedded function as a component in the real-time simulation of discrete-time predictive control system, which has been connected with other components. Specifically this function as shown in the figure has the two input variables u_{delta} and y , and the output variable $state_o$. We can apply the same technique to build other SIMULINK embedded functions from the MATLAB functions we had before.

Tutorial 10.2. This tutorial is to show how to build the embedded function for predictive control with constraints. Because the dimensions of some of the variables are required to be specified, for simplicity, we consider the case with four constraints and $N = 3$. The augmented system has two states. The dimensional numbers need to be checked, because if they are wrong, the program will not work. This is the main difference between the MATLAB program and the embedded SIMULINK program. All internal, local variables need to be defined explicitly. Hildreth’s quadratic programming is used in the computation.

Step by Step

1. Create a new embedded SIMULINK function called *DMPC_con.mdl*. Define the embedded function's input and output variables and parameters by entering the following programs into the file:

```
function [output_c,u_delta]=DMPC_con(set_point,X_hat,
phi_r,phi_x,L0,last_ucon,u_max,u_min,A_cons,b_d,Omega)
```

2. Go to 'Tool' and edit the data/ports. Output variables are *output_c* and *u_delta*; the input variables are *setpoint*, *Xhat*, *lastucon*; and the rest of the elements are parameters. Choose these specifications from the window given in data/ports.
3. Since Hildreth's quadratic programming is used in the computation, there are several internal, local variables. Unlike the MATLAB programs we wrote before, in the embedded function, these local variables are required to be declared. For this reason, we need to specify the number of constraints considered. There are only four constraints: u_{min} , u_{max} , Δu_{min} and Δu_{max} . When more constraints are included, the dimensions will be changed to suit the new specifications. As before, the dimensions of the parameter vectors are given as an example. Continue entering the following program into the file:

```
persistent con_val
if isempty(con_val)
con_val = zeros(1,1);
end
persistent b_a
if isempty(b_a)
b_a=zeros(2,1);
end
persistent P
if isempty(P)
P=zeros(4,4);
end
persistent d
if isempty(d)
d=zeros(4,1);
end
persistent kk
if isempty(kk)
kk=0;
end
persistent u_delta_ext
if isempty(u_delta_ext)
u_delta_ext=zeros(1,1);
end
```

```

persistent eta_ext
if isempty(eta_ext)
eta_ext=zeros(3,1);
end

```

4. A_{cons} , b are the pair of data matrices that define the constraints via

$$A_{cons} \times \eta \leq b.$$

So, we define b and calculate η , as well as check whether the constraints have been satisfied. Continue entering the following program into the file:

```

b_a=[u_max-last_ucon(1);-u_min+last_ucon(1)];
b=[b_d;b_a];
eta_1=phi_r*set_point; eta_2=phi_x*X_hat(1:2,1);
eta_ext=eta_1-eta_2;
kk=0; for ii=1:4
if (A_cons(ii,:)*eta_ext>b(ii)) kk=kk+1;end
end

```

5. If all constraints are satisfied, then $kk = 0$, the optimal solution is found. Continue entering the following program into the file:

```

if (kk==0)
u_delta_ext=L0'*eta_ext;
con_val=con_val+u_delta_ext;
end

```

6. If one or more constraints are violated, then we use Hildreth's quadratic programming procedure to find the active constraints. Continue entering the following program into the file:

```

%condition entering to the computation
%quadratic programming begins:
if (kk~=0)
P=A_cons*(Omega\A_cons');
d=(A_cons*(-eta_ext)+b);
lambda=zeros(4,1);
al=10;
for km=1:38
lambda_p=lambda;
for il=1:4
w= P(il,:)*lambda-P(il,il)*lambda(il,1);
w=w+d(il,1);
la=-w/P(il,il);
lambda(il,1)=max(0,la);
end
al=(lambda-lambda_p)'*(lambda-lambda_p);

```

```

if (al<10e-8);
break;
end
end

```

7. The Hildreth's programming procedure has found the optimal λ . If the i th constraint is active, $\lambda_i > 0$; otherwise, $\lambda_i = 0$ which shows that the constraint is inactive. Therefore, we can calculate the constrained solution directly using A_{cons} and the optimal λ found by the iterative computation. Continue entering the following program:

```

eta_ext=eta_ext-Omega\A_cons'*lambda;
%%%%%quadratic programming ends%%%%%
u_delta_ext=L0'*eta_ext;
con_val=con_val+u_delta_ext;
end

```

8. Finally define the output variables from this embedded function. Continue entering the following program into the file:

```

output_c = con_val;
u_delta=u_delta_ext;

```

Figure 10.8 shows the configuration of the discrete-time model predictive control system with constraints. It is seen in this figure that the embedded MATLAB functions written in this tutorial are used in the configuration, where the output of the MPC function is passed to the LabView I/O board as the control signal and the motor output is measured by the sensor and passed back by the LabView I/O to the embedded function of the observer.

10.3.3 Constrained Control of DC Motor Using xPC Target

The discrete-time model predictive control system is tested by controlling a DC motor, which is a teaching apparatus. The experiment setup is shown in Figure 10.9.

The control variable is voltage and the output variable is speed. The system is sampled with the sampling interval of 0.02 sec. A first-order model is obtained after a step test to describe the dynamics of the model as

$$x_m(k+1) = 0.8817x_m(k) + 0.5u(k); \quad y(k) = 0.7512x_m(k).$$

The augmented model with integrator is:

$$A = \begin{bmatrix} 0.8817 & 0 \\ 0.6623 & 1.0000 \end{bmatrix}; \quad B = \begin{bmatrix} 0.5 \\ 0.3756 \end{bmatrix}; \quad C = [0 \ 1].$$

The parameters for the cost function are selected as $Q = C^T C$; $R = 6$, $N_p = 16$. With the exponential weight factor of $\alpha = 1.2$ being used in the

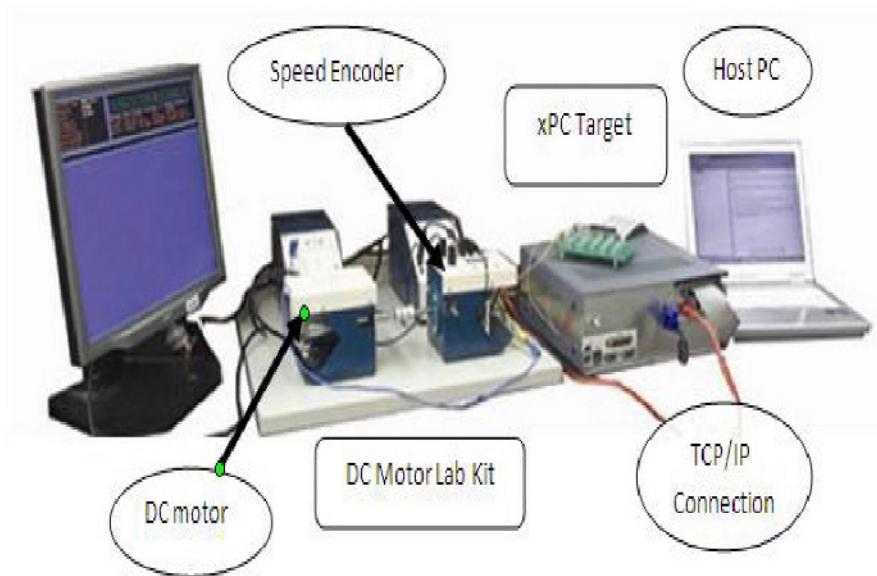


Fig. 10.9. Laboratory setup for controlling a motor using xPC target

design, with the solution of the algebraic Riccati equation, P , the modified weight matrices are

$$Q_\alpha = \begin{bmatrix} 2.1122 & 1.0472 \\ 1.0472 & 1.8120 \end{bmatrix}; \quad R_\alpha = 4.1667.$$

The parameters in the Laguerre functions are selected as $N = 3$; $a = 0.5$. With all the design parameters specified, the cost function for the discrete-time predictive control system is defined by

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(k_i) + \text{constant},$$

where

$$\Omega = \begin{bmatrix} 24.2686 & 16.8770 & 11.1140 \\ 16.8770 & 23.4915 & 15.6802 \\ 11.1140 & 15.6802 & 21.6471 \end{bmatrix} \quad \Psi = \begin{bmatrix} 21.1042 & 7.2411 \\ 15.5525 & 4.0584 \\ 9.7059 & 2.1970 \end{bmatrix}.$$

To avoid matrix inversion, the parameter vector η for the Laguerre function is calculated as

$$\eta = \phi_r \times r(k_i) - \phi_x \times x(k_i)$$

where the gain matrices are

$$\phi_x = \Omega^{-1}\Psi = \begin{bmatrix} 0.8985 & 0.3760 \\ 0.2562 & -0.0530 \\ -0.0813 & -0.1291 \end{bmatrix},$$

and ϕ_r is the last column of ϕ_x , which is

$$\phi_r = [0.3760 \ -0.0530 \ -0.1291].$$

The constraints are specified as $-0.6 \leq u(k) \leq 0.6$; $-0.1 \leq \Delta u(k) \leq 0.1$. The observer is designed using the pole-assignment method, where the closed-loop poles are placed at $\lambda_1 = 0.85$, $\lambda_2 = 0.86$ to obtain the observer gain $K_{ob} = [0.0010 \ 0.1717]^T$. All these data are stored in MATLAB workspace, in addition to the information required for the formulation of the constraints.

Figures 10.10a to 10.10c show the comparison of the closed-loop control experimental results with, and without constraints. Figure 10.10b shows that the imposed constraints have limited the peaks of the control amplitude, while Figure 10.10c shows the significant reduction on the peaks of the incremental control signal. The constraints on $\Delta u(k)$ and $u(k)$ limit the peak current consumption of the motor, and also provide protection to gear box. It is interesting to note from Figure 10.10a that the closed-loop response speed is almost identical for the two cases, except that the overshoot in the constrained control case is reduced. The experiment shows that the closed-loop control performance has negligible deterioration, while all hard constraints are met.

10.4 Control of Magnetic Bearing Systems

Active magnetic bearing (AMB) system is a typical mechatronic system. It is composed of mechanical components combined with electronic elements such as sensors, power amplifiers and controller. As compared to the characteristics of mechanical and hydro-static bearings, active magnetic bearings exhibit a dramatic reduction in friction, which allows efficient operation at high speed. The contactless nature also brings about other advantages such as the elimination of lubrication and less vibration. In industry, AMB can be applied in many areas, such as gas turbine engines, turbo-molecular vacuum pumps, generators, and linear induction motors.

However, an AMB system requires high-performance feedback control for operation since it is an open-loop unstable dynamical system. To apply effective control to the AMB system, it is necessary to construct a high performance, real-time feedback controller to suspend and stabilize the rotor and this has received much attention recently. The laboratory experimental system is an apparatus called MBC500.

The MBC500 magnetic bearing system includes a rotor shaft, two active magnetic bearing actuators and turbine driven by compressed air. On both ends of the actuators, hall effect sensors are placed to detect the displacements of the rotor shaft. With the internal analog controller, the rotor can be

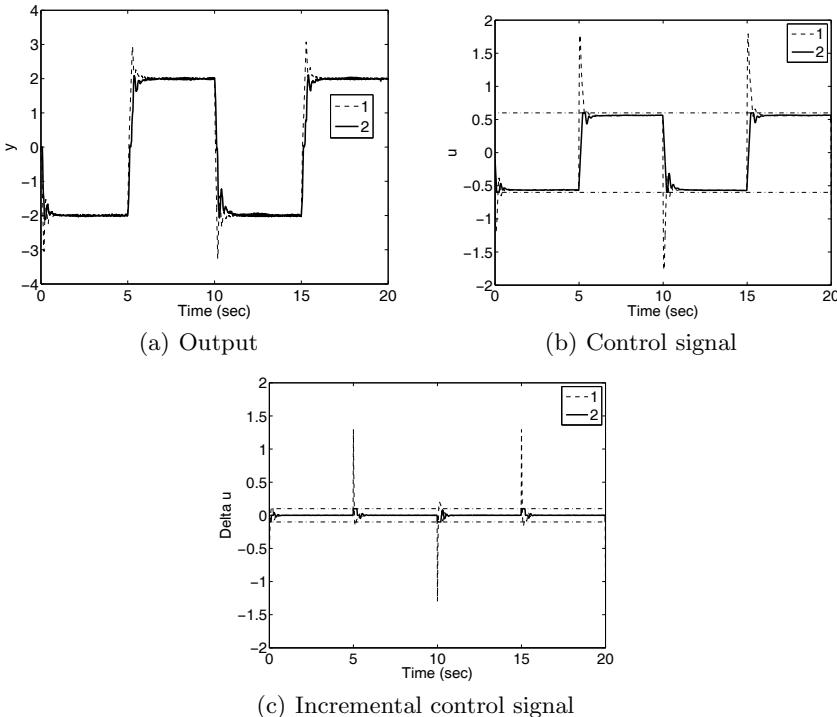


Fig. 10.10. Comparison of closed-loop control experimental results. Key: line (1) without constraints; line (2) with constraints ($-0.6 \leq u(k) \leq 0.6$, $-0.1 \leq \Delta u(k) \leq 0.1$)

levitated with no physical contact to surrounding magnets, and the rotational speed of the rotor shaft can reach up to 10000 rpm. There exist couplings in the horizontal or vertical direction. However, these are small and neglected for simplicity. The two channels for rotational vertical direction are held by the analogue controllers and the two channels in the horizontal directions are to be controlled by discrete-time model predictive controllers. Figure 10.11 shows the apparatus.

In this application, xPC target is used as the implementation platform for this real-time control system using standard PC hardware. The system is simulated by SIMULINK and MATLAB, and compiled to C code by Real-time Workshop, then transmitted into xPC target. Once completed, the xPC target becomes a stand-alone system to apply the control algorithm to the apparatus. Figure 10.8 shows the configuration of the real-time predictive control system via SIMULINK blocks, which is identical to the predictive control system structure used in controlling the motor, except that the variables have dimensions appropriate to the variables in the magnetic bearing apparatus.

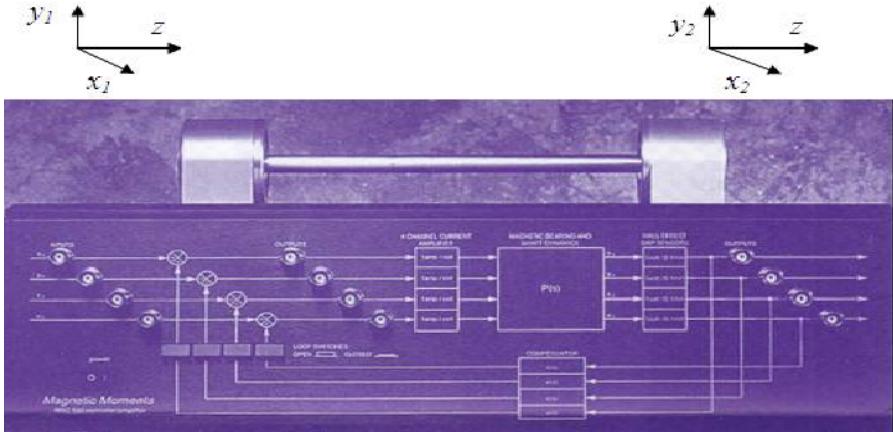


Fig. 10.11. The MBC500 magnetic bearing system

10.4.1 System Identification

After neglecting the couplings between the horizontal or vertical directions, the AMB system is regarded as two separate systems in both horizontal and vertical directions. Identification experiments are employed in the 2-input 2-output subsystem along the x-axis (horizontal) with the y-axis (vertical) controlled by built-in analogue controllers. Two uniformly distributed random number signals of different seeds are applied to two inputs in the form of voltage inputs. This system is sampled with the interval of 0.2 ms. The PEM (prediction error method) function in System Identification Toolbox of MATLAB is used to identify the plant model in discrete-time state-space form.

Here, the input variables are u_1 and u_2 ($u = [u_1 \ u_2]^T$), where u_1 is the voltage used to control the horizontal axis x_1 and u_2 is to control the horizontal axis x_2 . The two outputs are y_1 and y_2 ($y = [y_1 \ y_2]^T$), where y_1 is the horizontal displacement at the left end of the apparatus (termed x_1 in Figure 10.11), and y_2 is the displacement at the right end (termed x_2 in Figure 10.11). In the identification process, the model order is carefully selected so as to produce a model structure that is controllable and observable, and the state variables are not directly related to the physical variables in the system. Minimization of the sum of squared prediction errors is used in the selection of model structure.

The state-space discrete-time model with sampling interval 0.2 ms is identified as

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k) \\ y(k) &= C_m x(k), \end{aligned} \quad (10.6)$$

where the matrices are

$$A_m = \begin{bmatrix} 4.3875 & 6.0779 & 0.5034 & -0.0174 \\ -1.8758 & -2.3667 & -0.2701 & -0.6275 \\ -0.1295 & -0.2413 & 0.8758 & 7.9509 \\ 0.0004 & -0.0000 & -0.0005 & 1.0944 \end{bmatrix}; B_m = \begin{bmatrix} 0.0310 & -0.0000 \\ 0.0541 & -0.0000 \\ 0.0043 & -0.0000 \\ 0.0000 & 0.0625 \end{bmatrix}$$

$$C_m = \begin{bmatrix} 0.0265 & 0.0785 & 0.0009 & -0.0901 \\ -0.0491 & -0.0733 & -0.0240 & 0.1109 \end{bmatrix}.$$

The identified model is augmented to introduce integral action. The augmented model is denoted as (A, B, C) .

10.4.2 Experimental Results

The number of terms used in the Laguerre model to capture the 2-input signals is selected as $N_1 = 3$ for number one input, and $N_2 = 3$ for the number two input. The prediction horizon for DMPC is 160 steps forward. The pole of Laguerre functions are selected as $a_1 = a_2 = 0.95$. The weight matrices $Q = C^T C$, and $R = 0.1I$, where I is the identity matrix with appropriate dimension. Here the small weight 0.1 is used in R to produce a fast closed-loop response speed. The observer is designed using pole assignment, based on the pair (A^T, C^T) by allocating the observer poles at $[0.9210 \ 0.9220 \ 0.9230 \ 0.9240 \ 0.9250 \ 0.9260]$. As there is some measurement noise in the system, we found that it is best to keep the observer poles away from the origin of the complex plane so as to avoid amplification of the noise.

Figure 10.12 illustrates the step responses on horizontal axes x_1 and x_2 . The measured output is voltage, which is equivalent to the displacement of the rotor in each channel. Where r is the step reference trajectories, y_1, y_2, u_1 and u_2 are the corresponding outputs and control signals, respectively. There are two experiments that have been conducted as shown in Figure 10.12. The plots in the left column show the experimental results when there is a step change of 0.3 unit applied to the channel y_1 , and the set-point signal to y_2 remains unchanged. We can see from these figures that the closed-loop control system is oscillatory, however, the set-point change has been achieved without steady-state errors. Similarly, the plots in the right column of Figure 10.12 show the experimental results when there is a step change of 0.3 unit applied to the channel y_2 and the set-point signal to y_1 remains unchanged. The closed-loop response is also oscillatory.

Operational constraints were attempted in the predictive control system, however, the closed-loop became unstable as soon as the constraints became active. As the open-loop system is unstable with poles outside the unit circle, also on the unit circle, it is very difficult to control this system with constraints. Successful experiments have been conducted with constraints imposed in the outer-loop control system in which the unstable bearing is stabilized with this inner-loop predictive controller (Yang, 2008).

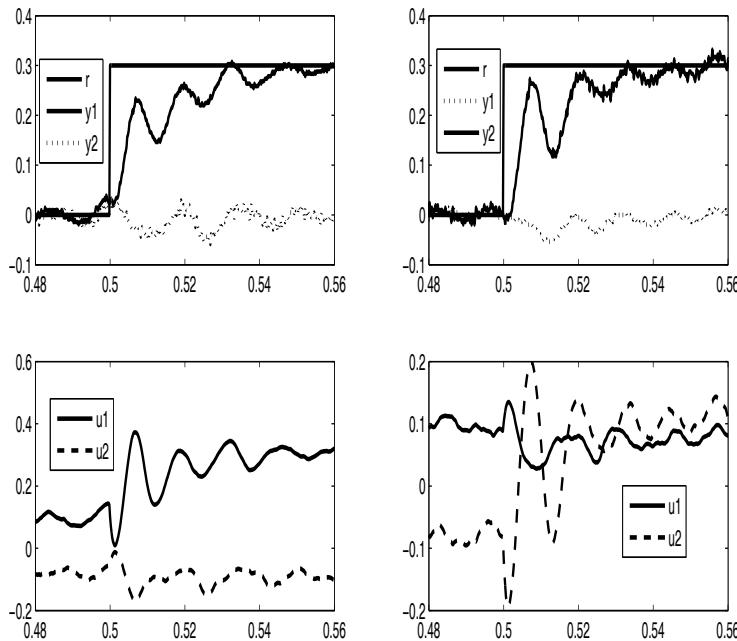


Fig. 10.12. Step response for channel y_1 and y_2 with $N_1 = N_2 = 3$, $a_1 = a_2 = 0.95$. The left column is for step response on channel y_1 , the right column is for a step response on channel y_2

10.5 Continuous-time Predictive Control of Food Extruder

Extrusion is a continuous process in which a rotating screw is used to force the food material through the barrel of the machine and out through a narrow die opening. In this process the material is simultaneously transported, mixed, shaped, stretched and sheared under elevated temperature and pressure. The extruder in the study is an APV-MPF40 co-rotating twin-screw extruder (see Figure 10.13). The block diagram of the food cooking extruder is shown in Figure 10.14. The food extruder uses ‘Gravimetric’ as its feeding system with which the feeding material enters the machine from the top part of the extruder. When it is in normal operation, the throughput is between 20 to 75 kg/hr. The screw diameter is 40 mm and its length to diameter ratio is 25:1. The experiments presented in this section were conducted using a screw profile that has a high shear configuration. A real-time, PC-based process monitoring and supervisory control system interfaces with the programmable logic controller (Siemens 95U PLC) in the extruder. This system continuously



Fig. 10.13. Twin-screw food extruder

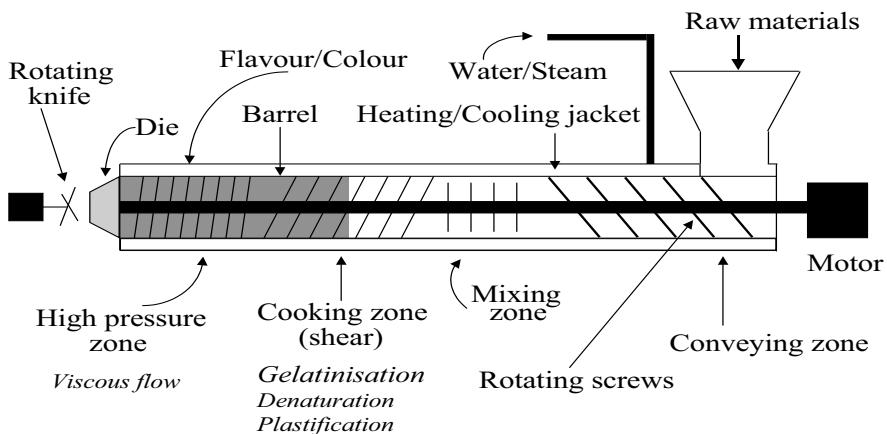


Fig. 10.14. Block diagram of a twin-screw food extruder

monitors and logs the temperatures, screw speed, feed rate and liquid injection rate. The system is sampled in one second intervals. The extruder has nine cooking zones, each of them has a set-point value for the temperature and an actual measured value. These zone temperatures are under PID control. There are also independent temperature measurements throughout the barrel

called melt temperature. Unlike the zone temperature variables, they are not controlled variables. If the location of the independent measurement (melt) is close to a particular zone, then the melt temperature should be close to the zone temperature variables, however, it may not be identical. There are 37 on-line measured variables in the food extruder, which are logged for analysis. These variables include screw speed, motor torque, SME, die pressure, cutter speed, material feed rate, liquid pump speed, the zone temperature measurements for the nine zones and the melt temperature measurements for the corresponding zones. Among them, the specific mechanical energy, SME, is calculated based on the relation:

$$SME = \frac{N_s \times T_t \times kW}{R_f}, \quad (10.7)$$

where N_s is the screw speed as % of maximum, T_t is the torque as % of maximum, kW is the motor power and R_f is the feed rate. The factors in a food extruder that can be adjusted on-line include the screw speed, the rate at which the raw material enters the extruder (feed rate) and the liquid injection rate (measured via liquid pump speed). Among the on-line measured variables, a previous study (Wang *et al.*, 2001) has found that motor torque, SME, die pressure, Melt M9 (melt temperature at the 9th zone) have a strong impact on the product quality attributes such as bulk density and moisture content. The primary manipulated variables for controlling these process variables are screw speed, liquid injection rate and material feed rate. Because in an industrial setting the material feed rate is often maintained as a constant, even though it has a strong influence on the process variables, it is avoided as a manipulated variable. In this work, screw speed and liquid injection rate are used as the manipulated variables, and specific mechanical energy and motor torque are used as the controlled variables.

10.5.1 Experimental Setup

The physical setup for the experiment is shown in Figure 10.15. The sensors fitted to the extruder are used to monitor a wide variety of variables including screw speed, motor torque, liquid pump speed, die pressure, product temperature and feed rate. Specific mechanical energy (SME) as stated before is a variable computed using screw speed, motor torque, motor power and feedrate. The programmable logic controller (PLC) serves to store temporarily the sensory data while providing low-level control functions for the extruder. The high-level data acquisition function is performed by a supervisory control and data acquisition (SCADA) system (The system is called Citect in this application). Data are transferred between the PLC and the main computer running the SCADA program via a serial cable link. The computer control system which integrates the PLC in a serial link with a SCADA system has been widely used for monitoring and controlling small-to-medium size industrial

processes, which include food extrusion processes. The real-time model predictive control system (developed for this application and is called ConQualEx Control) communicates with the SCADA system via its communication protocol. The communication protocol allows the user to assign tag numbers to the input and output variables. In this application, there are three input and three output variables used in the communication protocol. The variable tags read by the control module from the SCADA database are as follows:

1. *Screw – Speed – Set* (screw speed);
2. *Liq – pump – set* (liquid pump speed);
3. *Feed – Rate – Set* (powder feed rate);
4. *SME – tag* (specific mechanical energy);
5. *Motor – Torque* (motor torque);
6. *Die – Pressure* (die pressure).

The model predictive control module writes to the process inputs given through variable tags 1, 2, and 3. Similarly, the control module also reads the process outputs identified by variable tags 4, 5 and 6. The model predictive controller that calculates the optimal control movement stands alone in the main computer and is coded in C++ programming language. Although there are three inputs and three outputs used in the communication channels, the control signals are screw speed and liquid pump speed. The output signals used in the feedback control configuration are Specific Mechanical Energy (SME) and Motor Torque. The results in Wang *et al.*, (2001) demonstrated that a pair of product quality attributes (moisture content and bulk density) can be inferred from specific mechanical energy and motor torque.

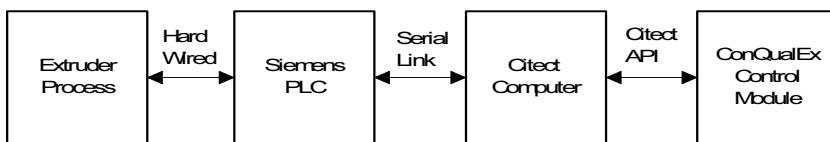


Fig. 10.15. Experimental setup of the real-time control system

10.5.2 Mathematical Models

A project on the system identification of the food extruder (Wang *et al.*, 2004) was performed in parallel to the model predictive control work presented here. In the identification project, a multi-frequency relay feedback control system (Wang and Cluett, 2000) was implemented on the food extruder to ensure safe operation of the process when performing the identification experiment and to obtain experimental data that has the relevant frequency content for dynamic modelling. Continuous-time transfer function models were estimated using the state variable filter approach developed by Wang and Gawthrop (2001). Some new results on identification of a continuous-time model can be found in Garnier and Wang (2008). More specifically, suppose that u_1 , u_2 , y_1 and y_2 represent screw speed, liquid pump speed, SME and motor torque respectively. Then, the continuous-time model for the food extruder is

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad (10.8)$$

where

$$\begin{aligned} G_{11} &= \frac{0.21048s + 0.00245}{s^3 + 0.302902s^2 + 0.066775s + 0.002186} \\ G_{12} &= \frac{-0.001313s^2 + 0.000548s - 0.000052}{s^4 + 0.210391s^3 + 0.105228s^2 + 0.00777s + 0.000854} \\ G_{21} &= \frac{0.000976s - 0.000226}{s^3 + 0.422036s^2 + 0.091833s + 0.003434} \\ G_{22} &= \frac{-0.000017}{s^2 + 0.060324s + 0.006836}. \end{aligned}$$

The continuous-time transfer function models have been validated using four sets of experimental data that are independent of the data sets used for estimating the models. The mathematical models are further validated in the design and implementation of the predictive control as demonstrated in the following.

The continuous-time transfer function model (10.8) is converted to a continuous-time state-space model using the MATLAB script given as below, where the multi-input and multi-output transfer function is defined with the four pairs of numerator and denominator, followed by converting it to a minimal state-space realization:

```
num11=[0.21048 0.00245];
den11=[1 0.302902 0.066775 0.002186];
num12=[-0.001313 0.000548 -0.000052];
den12=[1 0.210391 0.105228 0.00777 0.000854];
num21=[0.000976 -0.000226];
den21=[1 0.422036 0.091833 0.003434];
num22=[-0.000017]; den22=[1 0.060324 0.006836];
```

```

Gs=tf({num11 num12; num21 num22},
{den11 den12; den21 den22});
sys=ss(Gs, 'min');
[Am,Bm,Cm,Dm]=ssdata(sys);

```

Here a minimal realization is used to ensure that the state-space model is observable and controllable. The state-space model with minimal realization and 12 state variables is expressed as:

$$\begin{aligned}\dot{x}_m(t) &= A_m x_m(t) + B_m u(t) \\ y(t) &= C_m x_m(t)\end{aligned}\quad (10.9)$$

which is then used as the basis for the continuous-time model predictive controller design.

10.5.3 Operation of the Model Predictive Controller

The model predictive control system has a user interface to enable engineers to enter operational conditions and design parameters. A screen capture of the main control module window is shown in Figure 10.16. This window provides a means to access all other windows related to the model predictive control module. The windows that form part of the control module are the main control window; the model window; the control parameters window; and the observer window. When the real-time control program is started, the operational window is displayed. The control module may be accessed from this screen via the ‘Control’ menu on this screen.

SCADA Communications

The first step in opening communications is to ensure the SCADA system is running with the correct project. Once this has been checked, the user may select the ‘Open’ Button in the Communications group in the main control window. The status indicator in the Communication group should change from ‘Closed’ to ‘Open’ indicating that communications have been established.

Saturation Limits

The hard constraints on the control variables are realized through saturation mechanisms, which assumed that once the constraints were violated, they became active constraints. If the process input signal provided by the controller is beyond the input variables’ safe operating range or the safe rate of change, the input signal to the process is ‘clipped’ to the limiting value. Saturation limits can be adjusted in the Saturation block shown in Figure 10.16.

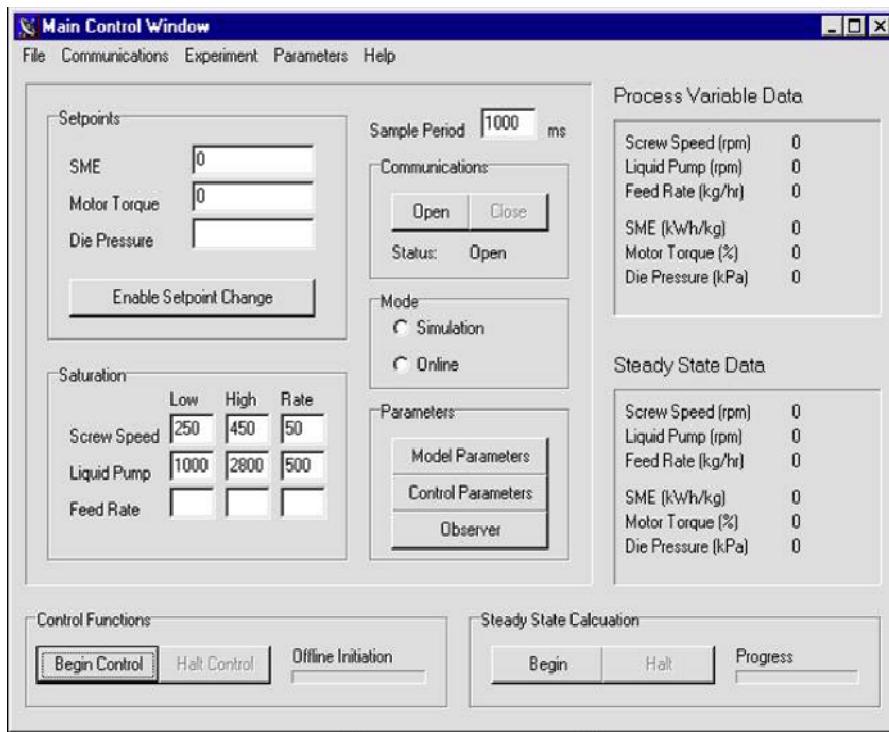


Fig. 10.16. Main operational window

Initiating On-line Control

Once all relevant control parameters are set, the ‘Begin Control Experiment’ button is enabled. When this button is pressed the program performs all off-line calculations. Once completed, the on-line phase is initiated and the process is under automatic control. The ‘Halt Control Experiment’ button becomes enabled, and it may be pressed at any point thereafter to stop the on-line phase of the experiment.

10.5.4 Controller Tuning Parameters

The tuning parameters in the continuous-time model predictive control algorithm are parameter p , which is the scaling factor in the Laguerre functions describing derivative of the control signal; parameter N , which is the number of terms used in the Laguerre functions; the prediction horizon parameter T_p ; and the weighting matrices Q and R . These parameter settings in the experiments are discussed as below.

Parameter p

In the extruder control case, the scaling factor p for screw speed is chosen to be $p_1 = 1/30$ and for liquid pump speed $p_2 = 1/15$. Both scaling factors are selected to be approximately the same size as the open-loop dominant poles.

Parameter N

The parameter N is the number of terms that are used in capturing the future control signal. As N increases, the degrees of freedom in describing the control trajectory increase and the control signal tends to the underlying optimal control defined by the choice of Q and R matrices. In the extruder control case, the number of terms used for describing both control signal trajectories is chosen to be $N_1 = N_2 = 3$.

Prediction Horizon T_p

For a stable system, the prediction horizon is recommended to be chosen approximately equal to (or greater than) the open-loop process settling time. The prediction horizon T_p is chosen to be 300 sec, which is greater than the open-loop settling time.

The Weighting Matrices

The weighting matrices in the cost function of MPC are chosen to be $Q = C^T C$ and $R = I$, where I is the (2×2) identity matrix. Increasing the size of the components in the R matrix will decrease the closed-loop response speed.

Observer Gain

The observer is designed using a pole-assignment strategy by placing the observer's dominant closed-loop pole at $-2/10$, which is about twice the size of the largest scaling factor of the Laguerre polynomial, and the rest of the observer poles are placed on the locations of $-2/10 - k * 0.5$ where $k = 2, 3, \dots, 14$.

10.5.5 On-line Control Experiments

The food extruder is brought to a steady-state operational condition before the control system is switched on. Experiments are carried out for regulatory control and set-point changes.

Steady-State Operating Conditions

The feed material used in the control experiments is maize flour and the product is snack chips. The process steady-state operating condition is determined by the following parameters: Cutter Speed = 1620 rpm, Steady State Screw Speed = 300 rpm, Steady State Liquid Pump Speed = 2000 rpm and Material Feed Rate = 20 kg/hr. The corresponding steady-state output variables are calculated as SME = 639 kWh/kg and Motor Torque = 42.0%. The screw speed is constrained to operate between 250 rpm to 400 rpm with a maximum allowable change of 50 rpm. The liquid pump speed is constrained to operate between 1000 rpm to 2800 rpm with a maximum allowable change of 500 rpm. The control experiments conducted naturally satisfied the specified process constraints. A number of different experiments were carried out to assess the performance of the controller, including regulatory control, and set-point changes for each process output.

The results from these experiments are discussed as follows.

Regulatory Control

The food extruder was brought to the steady-state operational condition by applying constant screw speed, liquid pump speed and material feed rate. Once the process had settled, data on the process variables were gathered. The steady-state value in the process output was $y_{ss} = [SME \ MT]^T = [639 \ 42]^T$ where SME and MT denote steady-state values of SME and motor torque. In addition, the steady-state values of screw speed and liquid pump speed were measured as $u_{ss} = [SS \ LPS]^T = [300 \ 2000]^T$. At the process steady state operation (say time t_i), set the initial conditions of the estimated state variable $\hat{X}(t_i) = 0$, and the incremental process output variable $y(t_i) = [0 \ 0]^T$. The previous control signal to the food extruder is initialized as $u(t_i - \Delta t) = u_{ss}$. The implementation of the predictive control system is performed using the following steps.

1. Calculate $(r(t_i) = 0)$

$$\eta = -\Omega^{-1}\{\Psi\hat{x}(t_i)\} \quad (10.10)$$

$$\dot{u}(t_i) = [L_1(0)^T \ L_2(0)^T] \eta \quad (10.11)$$

$$u(t_i) = u(t_i - \Delta t) + \dot{u}(t_i)\Delta t. \quad (10.12)$$

2. Calculate the estimated state variable $\hat{x}(t_i + \Delta t)$, using

$$\hat{x}(t_i + \Delta t) = \hat{x}(t_i) + \{A\hat{x}(t_i) + B\dot{u}(t_i) + K_{ob}(y(t_i) - C\hat{x}(t_i))\}\Delta t. \quad (10.13)$$

3. Write the control signal to Variable tags 1 and 2 through the SCADA Communication Protocol.
4. Read the actual plant output y_{act} via Variable tags 4 and 5, and input u_{act} via the Variable tags 1 and 2, and calculate $y(t_i + \Delta t) = y_{act} - y_{ss}$; $u(t_i) = u_{act}$.

5. Go to Step 1 with one increment of Δt and the update on $\hat{x}(t_i)$, $u(t_i - \Delta t)$ and $y(t_i)$.

The above computational algorithm generated the values of screw speed and liquid pump speed for the closed-loop operation of food extruder. Plots of the process variables for the regulatory control result are shown in Figure 10.17. The top two plots in Figure 10.17 show the process outputs, *i.e.*, the SME and motor torque, while the bottom two plots show the manipulated variables, *i.e.*, the screw speed and liquid pump speed. It is seen that the plant operation made a smooth transition from open-loop operation to closed-loop operation. The liquid pump speed, however, made an adjustment to a new steady state value with respect to the specification of the process output variables. Because the calculation of the control law is based on the derivatives of the control signal, it is seen here that it is relatively easy to make the transition from the open-loop operation to the closed-loop operation. In addition, the changes in the control signal are gradual and smooth. It is also seen from this figure that disturbances occur due to the changes in material feedrate, yet the continuous-time model predictive controller rejects the disturbance and maintains the process outputs at the specified operating conditions.

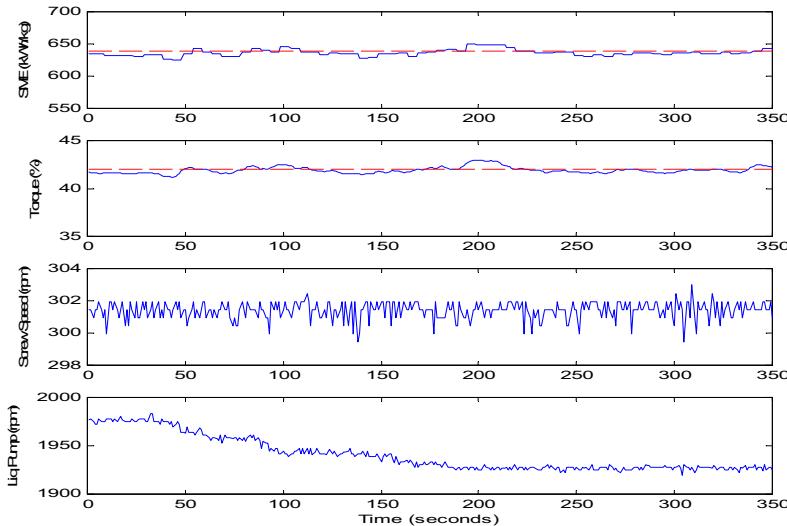


Fig. 10.17. Regulatory control of food extruder. Top two sub-figures: process outputs (SME and motor torque); Bottom two sub-figures: control signals (screw speed and liquid pump speed)

Reference Following Control

Two experiments were performed to examine the ability of the model predictive controller to automatically adjust the process outputs to new set-points. In the implementation of the set-point changes, the on-line control algorithm listed above applies except that Step One is changed into the following computational equation:

$$\eta = \Omega^{-1}\{\psi_r r(t_i) - \Psi \hat{x}(t_i)\}, \quad (10.14)$$

where ψ_r is the last two columns of matrix Ψ , and the set-point signal $r(t_i)$ was specified for the two cases.

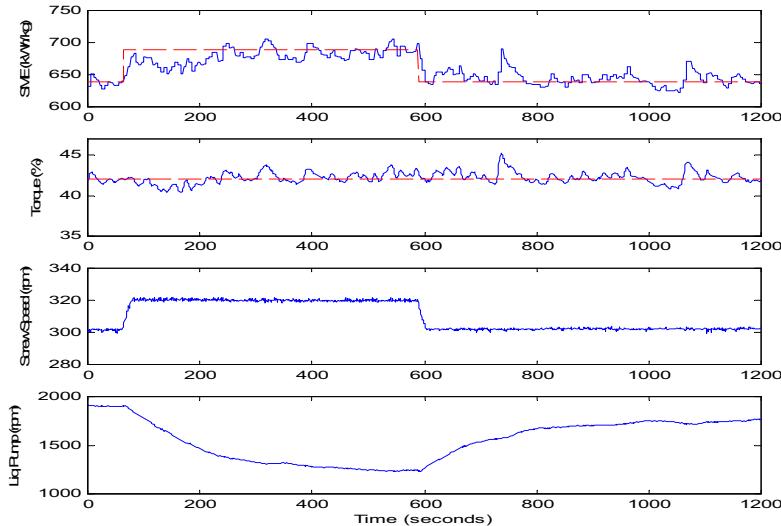


Fig. 10.18. Set-point change for SME. Top two sub-figures: process outputs (SME and motor torque); Bottom two sub-figures: control signals (screw speed and liquid pump speed)

Case A. Set-point change of SME

Figure 10.18 shows the experimental results for a set-point change of SME from 639 kWh/kg to 689 kWh/kg and back to 639 kWh/kg, while maintaining motor torque to be constant. The increment of the SME set-point signal is 50 kWh/kg for the first step change, then with the change of -50 kWh/kg,

the SME set-point signal is brought back to its original value of 639 kWh/kg. The incremental change of the motor torque is zero. More specifically, $r(t_i) = [0 \ 0]^T$ for $0 \leq t_i \leq 69$ sec, $r(t_i) = [50 \ 0]^T$ for $70 \leq t_i \leq 594$ sec and $r(t_i) = [-50 \ 0]^T$ for $595 \leq t_i \leq 1200$ sec. The plots in Figure 10.18 show the SME output, motor torque output, screw speed and liquid pump speed signals. It is seen that the SME output variable has been automatically adjusted to the new set-point within several minutes of the set-point change, then brought back to its original operational point. This has been achieved while maintaining the motor torque output close to its specified set-point.

Case B. Set-point change of motor torque

Case B investigates the set-point change of motor torque from 40.6% to 43.6% while maintaining SME as constant. The tuning parameters in the model predictive controller are selected as $p_1 = 1/30$, $p_2 = 1/10$, $Q = I$ and $R = I$. The increment of the motor torque set-point signal is 3%, and the increment of the SME set-point signal is zero. In this experiment, $r(t_i) = [0 \ 0]^T$ for $0 \leq t_i < 3$ sec and $r(t_i) = [0 \ 3]^T$ for $3 \leq t_i \leq 750$ sec. Figure 10.19 shows the experimental results obtained in Case 2. It is seen that the motor torque output variable has been automatically adjusted to the new set-point value,

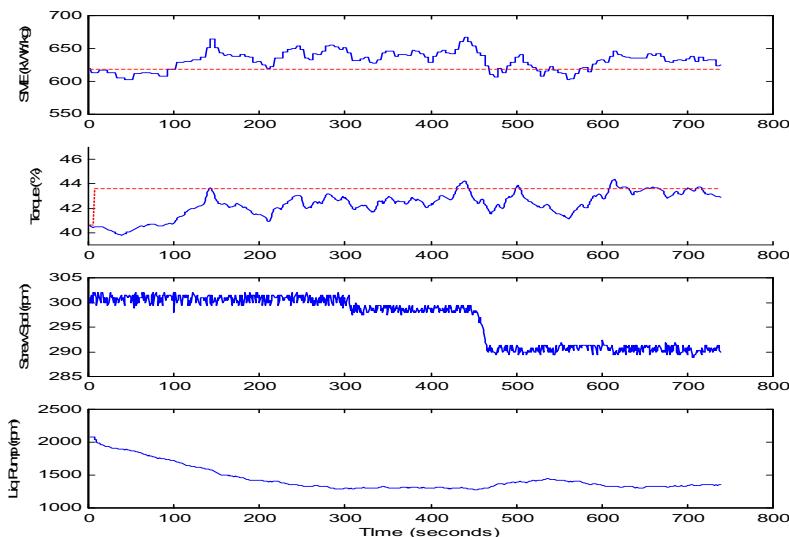


Fig. 10.19. Set-point change for motor torque. Top two sub-figures: process outputs (SME and motor torque); Bottom two sub-figures: control signals(screw speed and liquid pump speed)

while the SME output variable has been maintained close to its specified set-point value. The results also show that the change of motor torque affects the SME which is calculated partly based on the motor torque. After about 400 seconds, the predictive control system has brought the SME back to its original set-point and the motor torque to the new set-point.

10.6 Summary

This chapter has discussed the real-time implementation of model predictive control with constraints. The approaches used the MATLAB functions we developed in the previous chapters to generate the cost J in the form of quadratic function of decision variable η :

$$J = \eta^T \Omega \eta + 2\eta^T \Psi x(k_i) + \text{constant}.$$

where Ω and Ψ are computed off-line with the MATLAB function *dmpc.m* for discrete-time application and *cmpc.m* for continuous-time application. If it is a continuous-time application, $x(k_i)$ is replaced by $x(t_i)$. Also, if an observer is used in the implementation, then $x(k_i)$ is replaced by the estimate $\hat{x}(k_i)$ or the estimate $\hat{x}(t_i)$ in the continuous-time application. The constraints are also formulated off-line with appropriate dimensions and relationships. Some of the constraints are required to be updated on-line.

The first implementation uses a micro-controller for controlling a DC motor. The model predictive control system is designed using discrete-time NMSS model, so that the feedback signals are differenced input, output signals ($\Delta u(.)$ and Δy) and output signal $y(.)$. Because the micro-controller has limited capacity for computation, the constraints are imposed analytically by enforcing them on the first sample of both control and incremental of the control. The implementation procedure is simple and straightforward, whose complexity does not exceed that required in the implementation of a PID controller with anti-windup protection. The implementation is written in C language. Experimental results have shown that the control system has excellent performance.

The second implementation uses MATLAB/ SIMULINK xPC target as the platform. In this implementation, there are two computers involved: one is the host computer that we use to edit the programs and compile the programs; the other is the target computer that executes the compiled C program in real time as a controller. In this configuration, the control system has the capacity to perform on-line optimization. In this chapter we showed how to convert the MATLAB programs we developed in the previous chapters into MATLAB embedded programs for implementation of predictive control systems, which also includes Hildreth's quadratic programming procedure. One of the main differences when translating programs from MATLAB to embedded programs is that in the latter applications, the dimensions of the variables are required to be specifically defined, application by application. MATLAB/SIMULINK

xPC target provides a powerful platform for testing and implementing predictive control systems. The real-time predictive control systems with constraints were demonstrated using a DC motor and a magnetic bearing system.

The third implementation uses the platform of a real-time PC-based supervisory control and data acquisition (SCADA) system. A food extruder is controlled by the continuous-time model predictive controller with successful experimental tests.

References

1. F. Allgower, T. A. Badgwell, S. J. Qin, J. B. Rawlings, and S. J. Wright. Nonlinear predictive control and moving horizon estimation- an introductory overview. in *Advances in Control: Highlights of ECC'99* (editor: Paul M. Frank), London, Springer, pages 391–449, 1999.
2. B. D. O. Anderson and J. M. Moore. *Linear Optimal Control*. Prentice-Hall, Hemel Hempstead, 1971.
3. B. D. O. Anderson and J. M. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, New Jersey, 1979.
4. K. J. Astrom and T. Hagglund. *Automatic Tuning of PID Controllers*. Instrument Society of America, Research Triangle Park, NC, 1988.
5. K. J. Astrom and T. Hagglund. *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, Research Triangle Park, NC, 1995.
6. K. J. Astrom and B. Wittenmark. *Computer Controlled Systems: Theory and Design*. Prentice-Hall, Englewood Cliffs, N. J., 3rd edition, 1997.
7. J. S. Bay. *Fundamentals of Linear State Space Systems*. McGraw-Hill, 1999.
8. A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulators for constrained control. *Technical report AUT99-16, ETH Zurich*, 1999.
9. R. R. Bitmead, M. Gevers, and V. Wertz. *Adaptive Optimal Control the thinking man's GPC*. Prentice Hall, New York, 1990.
10. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
11. E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer, New York, 2004.
12. A. Chotai, P.C. Young, P.G. Mckenna, and W. Tych. Proportional-integral-plus (PIP) design for delta operator systems: Parts 1 and 2. *International Journal of Control*, 70:123–147, 149–168, 1998.
13. D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalized predictive control.part 1: The basic algorithm. part 2: Extensions and interpretations. *Automatica*, 23:137–160, 1987.
14. C. R. Cutler and B. L. Ramaker. Dynamic matrix control-a computer control algorithm. *Presented at the Meeting of the American Institute of Chemical Engineers, Houston, Texas*, 1979.

15. C.K. Finn, B. Wahlberg, and B. E. Ydstie. Constrained predictive control using orthogonal expansions. *AIChE Journal*, 39:1810–1826, 1993.
16. R. Fletcher. *Practical Methods of Optimization, Volume 2, Constrained Optimization*. John Wiley and Sons, New York, 1981.
17. G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Welsley Publishing Company, Second Edition, 1991.
18. C. E. Garcia and A. M. Morshedi. Quadratic programming solution to Dynamic Matrix Control (QDMC). *Chemical Engineering Communication*, 46:73–87, 1986.
19. C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice - a survey. *Automatica*, 25:335–348, 1989.
20. H. Garnier and L. Wang eds. *Continuous-time Model Identification from Sampled Data*. Springer-Verlag, 2008.
21. P. J. Gawthrop. Linear predictive pole-placement control: practical issues. *39th IEEE Conference on Decision and Control, Sydney, Australia*, 160–165, 2000.
22. P. J. Gawthrop and E. Ronco. Predictive pole-placement control with linear models. *Automatica*, 38:421–432, 2002.
23. P. J. Gawthrop and L. Wang. Intermittent predictive control of an inverted pendulum. *Control Engineering Practice*, 14:1347–1356, 2006.
24. P. J. Gawthrop and L. Wang. Intermittent model predictive control. *Journal of Systems and Control Engineering*, 221:1007–1018, 2007.
25. G. C. Goodwin, S. Graebe, and M. Salgado. *Control System Design*. Prentice-Hall, Englewood Cliffs, NJ, 2000.
26. V. Gopal and L. T. Biegler. Large scale inequality constrained optimization and control. *IEEE Control System Mag.*, 18:59–68, 1998.
27. M. J. Grimble and M. A. Johnson. *Optimal Control and Stochastic Estimation: Theory and Applications, Volume 1*. John Wiley and Sons, Chichester, U. K., 1988a.
28. M. J. Grimble and M. A. Johnson. *Optimal Control and Stochastic Estimation: Theory and Applications, Volume 2*. John Wiley and Sons, Chichester, U. K., 1988b.
29. A. Hansson. A primal-dual interior-point method for robust optimal control of linear discrete-time systems. *IEEE transac. on Auto. Control*, 45, 2000.
30. M. A. Johnson and M. H. Moradi. *PID Control: New identification and design methods*. Springer Verlag, New York, USA, 2005.
31. T. Kailath. *Linear Systems*. Prentice Hall, Inc., Englewood Cliffs, N.J., 1980.
32. W. H. Kautz. Transient synthesis in time domain. *IRE Transactions on Circuit Theory*, CT-1(3):29–39, 1954.
33. S. S. Keerthi and E. G. Gilbert. Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations. *Journal of Optimization Theory and Applications*, 57:265–293, 1988.
34. B. Kouvaritakis and J. A. Rossiter. Stable generalized predictive control. *IEE Proceedings, Part D*, 139:349–362, 1992.
35. W.H. Kwon and S. Han. *Receding Horizon Control- model predictive control for state models*. Springer, 2005.
36. Y. W. Lee. *Statistical Theory of Communication*. John Wiley and Sons, New York, 1960.
37. L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Englewood Cliffs, New Jersey, 1999.

38. D. G. Luenberger. *Optimization by Vector Space Methods*. John Wiley and Sons, New York, 1969.
39. D. G. Luenberger. *Linear and Nonlinear Programming*. Second edition, Addison-Wesley Publishing Company, 1984.
40. J. M. Maciejowski. *Predictive Control with Constraints*. Pearson Education Limited, 2002.
41. D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: stability and optimality. *Automatica*, 36:789–814, 2000.
42. H. Michalska and D. Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Trans. on Autom. Control*, 40:995–1006, 1993.
43. M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers and Chemical Engineering*, 667–682, 1999.
44. K. Muske and J. B. Rawlings. Linear model predictive control of unstable processes. *Journal of Process Control*, 3, 1993.
45. K. Muske and J. B. Rawlings. Model predictive control with linear models. *AICHE J.*, 39, 1993.
46. A. W. Ordys and D. W. Clarke. A state-space description for GPC controllers. *International Journal of Systems Science*, 24:1727–1744, 1993.
47. V. Peterka. Predictor-based self tuning control. *Automatica*, 20:39–50, 1984.
48. S. J. Qin and T. A. Badgwell. An overview of industrial model predictive control technology. In *Preprints of Chemical Process Control-CPC V, California*, 1996.
49. C. V. Rao, S. J. Wright, and J. B. Rawlings. On the application of interior point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99:723–757, 1998.
50. J. B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems Magazine*, 20:38–52, 2000.
51. J. B. Rawlings and K. R. Muske. The stability of constrained receding horizon control. *IEEE Trans. Auto. Control*, 38:1512–1516, 1993.
52. J. Richalet. Predictive functional control: application to fast and accurate robots. *IFAC 10th World Congress, Munich, PRG*, 1987.
53. J. Richalet. Industrial applications of model based predictive control. *Automatica*, 29:1251–1274, 1993.
54. J. Richalet. Industrial implementation of predictive control. *Plenary address, UKACC International Conference on Control, Cambridge, U.K.*, 2000.
55. J. Richalet, A. Rault, J. L. Testud, and J. Papon. Model predictive heuristic control: applications to industrial processes. *Automatica*, 14:413–428, 1978.
56. N. L. Ricker. Use of quadratic programming for constrained internal model control. *Ind. Eng. Chem. Process Des. Dev.*, 24:925–936, 1985.
57. N. L. Ricker. Model-predictive control: state of the art. *Proc. Fourth International Conference on Chemical Process Control, Padre Island, Texas*, 271–296, 1991.
58. J. A. Rossiter. *Model-based Predictive Control, a Practical Approach*. Prentice Hall Int, 2003.
59. J. A. Rossiter and B. Kouvaritakis. Constrained stable generalised predictive control. *IEE Proceedings, Part D*, Vol. 140:243–254, 1993.
60. J. A. Rossiter, B. Kouvaritakis, and M. J. Rice. A numerically stable robust state-space approach to stable-predictive control strategies. *Automatica*, 34:65–73, 1998.
61. M. M. Seron, J. A. De Dona, and G. C. Goodwin. Global analytical predictive control with input constraints. *39th Conference on Decision and Control*, 2000.

62. S. L. Shah. A tutorial introduction to constrained long range predictive control. *Pulp and Paper Canada*, 94:57–63, 1995.
63. T. T. C. Tsang and D. W. Clarke. Generalized predictive control with input constraints. *Proc. IEE, Part D.*, 135:415–460, 1988.
64. E. T. van Donkelaar, O. H. Bosgra, and P.M.J. Van den Hof. Constrained model predictive control with on-line input parameterization. In *Proc. the 38th Conference on Decision and Control, Phoenix, Arizona*, 3718–3721, 1999.
65. B. Wahlberg. System identification using Laguerre models. *IEEE Transactions on Automatic Control*, 36:551–562, 1991.
66. C. Wang and P.C. Young. Direct digital control by input-output, state variable feedback: theoretical background. *International Journal of Control*, 74:97–109, 1988.
67. L. Wang. Use of orthonormal basis functions in continuous-time mpc design. *UKACC*, 2000.
68. L. Wang. Discrete model predictive control design using Laguerre functions. *Proceedings of American Control Conference*, 2001a.
69. L. Wang. Continuous time model predictive control using orthonormal functions. *International Journal of Control*, 74:1588–1600, 2001b.
70. L. Wang. Use of exponential data weighting in model predictive control design. *Proceedings of the 40th IEEE Conference on Decision and Control*, 2001c.
71. L. Wang. Discrete model predictive control design using Laguerre functions: Numerical sensitivity analysis. *Proceedings of American Control Conference*, 2003.
72. L. Wang. Discrete model predictive control design using Laguerre functions. *Journal of Process Control*, 14:131–142, 2004.
73. L. Wang, C. Chessari, and E. Karpiel. Inferential control of product quality attributes: application to food cooking extrusion process. *Journal of Process Control*, 11:621–636, 2001.
74. L. Wang and W. R. Cluett. *From Plant Data to Process Control: Ideas for Process Identification and PID Design*. Taylor and Francis, London, 2000.
75. L. Wang, P. Gawthrop, C. Chessari, and T. Podsiadley. Continuous time system identification of food extruder: experiment design and data analysis. *Proceedings of IFAC Symposium on System Identification, Holland*, 2003.
76. L. Wang, P. Gawthrop, C. Chessari, and T. Podsiadley. Indirect approach to continuous time system identification of food extruder. *Journal of Process Control*, 14:603–615, 2004.
77. L. Wang and P. J. Gawthrop. On the estimation of continuous time transfer functions. *International Journal of Control*, 74:889–904, 2001.
78. L. Wang and G. C. Goodwin. Integrating identification with robust control: a mixed H_2/H_∞ approach. *39th IEEE Conference on Decision and Control, Sydney, Australia*, 2000.
79. L. Wang, G. C. Goodwin, and D. H. Owens. Robust control of unstable systems using statistical confidence bounds. *Proceedings of European Control Conference, September*, 1999.
80. L. Wang and P. C. Young. An improved structure for model predictive control using non-minimal state space realisation. *Journal of Process Control*, 16:355–371, 2006.
81. L. Wang, P. C. Young, P.J. Gawthrop, and C.J. Taylor. Non-minimal state space model-based continuous-time model predictive control with constraints. *To appear in International Journal of Control*, 2009.

82. D.A. Wismer and R. Chattergy. *Introduction to Nonlinear Optimization, a Problem Solving Approach*. North-Holland, New York, 1978.
83. C. R. Wylie. *Advanced Engineering Mathematics*. McGraw-Hill, New York, 1960.
84. H. Yang. *Model Predictive Control of Magnetic Bearing System*. Masters degree thesis, RMIT University, Australia, 2008.
85. T. W. Yoon and D. W. Clarke. Receding-horizon predictive control with exponential weighting. *Int. Journal of Systems Science*, 1745–1757, 1993.
86. P. C. Young, M.A. Behzadi, Wang C. L., and A. Chotai. Direct digital and adaptive control by input-output, state variable feedback pole assignment. *International Journal of Control*, 46:1867–1881, 1987.
87. P. C. Young, M.J. Lee, A. Chotai, W. Tych, and Z. S. Chalabi. Modelling and PIP control of a glasshouse micro-climate. *Control Eng. Practice*, 2(4):591–604, 1994.
88. E. Zafiriou. On the effect of tuning parameters and constraints on the robustness of model predictive control. *Proc. Fourth International Conference on Chemical Process Control, Padre Island, Texas*, 363–393, 1991.
89. C. C. Zervos and G. A. Dumont. Deterministic adaptive control based on Laguerre series representation. *Int. J. Control*, 48:2333–2359, 1988.
90. A. Zheng. Reducing on-line computational demands in model predictive control by approximating QP constraints. *Journal of Process Control*, 9:279–290, 1999.

Index

- active constraints, 58, 61, 72, 76, 121, 131
active magnetic bearing system, 349
active set methods, 60
Algebraic Riccati equation, 101, 278, 283, 348
algebraic Riccati equation, 33, 157, 159, 168, 169, 234
asymptotic stability, 158, 160, 277
augmented model, 5, 23, 96, 113, 135, 156, 212
closed-form solution, 131
closed-loop control system, 16, 103, 108, 228
closed-loop eigenvalues, 17, 35, 36, 103, 105, 158, 169, 177, 284, 308
closed-loop poles, 324
closed-loop simulation, 115, 116, 138
cmpc.m, 229, 280
computational delay, 264, 265
condition number, 19, 150, 156, 157, 161, 170, 181, 272, 290, 305
constrained optimal solution, 52, 53, 57
constraints on control amplitude variation, 48, 73, 122, 124, 136, 250, 255, 340
constraints on control incremental variation, 47, 70, 118, 121, 124, 136, 340
constraints on output, 48, 78, 139, 251
constraints on rate of change of control signal, 249, 252
continuous stirred tank reactor, 111
control horizon, 7, 26
control trajectory, 99, 100, 102
controllability, 24, 214, 215
controller tuning parameters, 359
convolution integral, 220
convolution sum, 98
cost function, 3, 9, 97, 107, 152, 168, 172, 257
coupled tanks, 146
cssimucon.m, 258
cssimuob.m, 237
DC motor, 136, 333, 347
decision variable, 53, 61
dependent constraints, 56
deterministic disturbance, 212, 304
Digital Signal Processor, 334
Discrete Linear Quadratic Regulator, 95, 101
distillation column, 268
DMC, 297, 326
dmpc.m, 108, 121, 124, 137
double integrated plant, 36, 150
double-integrated plant, 155, 160
DSP, 335
dual quadratic programming, 62, 68, 131
dual root locus, 309
dual variable, 62
eigenvalues, 23, 214, 276
equality constraints, 53
exponential data weighting, 152, 274, 279

- exponentially decreasing weighting, 152
- exponentially increasing weighting, 152
- exponentially weighted cost function, 153, 154, 156, 275
- exponentially weighted variables, 153, 179, 277, 288
- exptut.m, 280
- extmodel.m, 6
- feasible solution, 56, 57
- FIR, 327
- food extruder, 147, 353
- future state variables, 8, 26, 107
- gain margin, 311
- glasshouse, 315
- GPC, 298, 301
- Hessian matrix, 52, 150, 156, 157, 161, 163, 273
- Hildreth's quadratic programming procedure, 63, 67, 71, 266, 345
- Int.m, 223
- implementation filter, 323, 330
- implementation of receding horizon control, 20
- impulse response, 90, 93
- inactive constraints, 58, 121, 131
- inequality constraints, 58, 60, 61, 154, 257
- inside-the-unit-circle zeros, 177, 308
- integral action, 17, 23, 228
- Kalman filter, 33, 34
- Kautz functions, 218, 241, 242
- Kuhn-Tucker conditions, 58
- LabView, 336, 347
- lagc.m, 196, 230, 256
- lagd.m, 90, 120, 122, 137
- lagdmodel.m, 91
- Lagrange multipliers, 54, 58, 68, 76
- Laguerre coefficients, 94, 106
- Laguerre functions, 87, 89, 90, 94, 106, 218
- Laguerre model, 91, 106, 328
- Laguerre networks, 86, 106
- left matrix fraction description, 315
- LQR, 234, 239, 283
- Mder.m, 253
- Mdu.m, 119
- micro-controller, 335
- minimal realization, 24, 113, 215, 299
- minimum of cost function, 9, 12, 97, 174, 226
- motor1.m, 136
- motor2.m, 140
- mpcgain.m, 13
- Mu.m, 122
- Mucon.m, 256
- multi-input and multi-output systems, 22
- non-minimal state-space realization, 298, 315, 321
- Nyquist plot, 312
- objective function, 54
- observability, 24, 30, 214, 215
- observer, 27, 34, 235
- observer gain matrix, 28
- observer poles, 32, 34, 352
- optimal solution, 10, 51, 108, 154, 167, 257, 277
- optimization window, 3, 7, 26, 93, 165, 264
- orthonormal expansion, 219, 240
- orthonormal functions, 210, 217
- orthonormality, 86, 225
- performance tuning parameters, 165, 176, 286
- phase margin, 311
- predicted output vector, 8
- predicted state variables, 8
- prediction horizon, 3, 7, 26, 151, 272, 286
- prescribed degree of stability, 167, 169, 283, 284, 287
- primal variable, 62
- Pulse Width Modulations, 334
- QPhild.m, 67
- quadratic programming, 52, 53, 62, 259
- reced.m, 20
- receding horizon control, 3, 15, 27, 35, 98, 277
- regulatory control, 361

- SCADA, 358
scaling factor of Laguerre functions, 86, 119
set-point signal, 9, 96, 224, 235
SIMULINK embedded function, 342, 344
`simuob.m`, 117
`simuuc.m`, 115
state estimate predictive control, 35
state estimation, 27, 234, 352
state feedback control, 16, 105, 108, 161, 165, 273, 324
state-space model, 4, 28
stochastic disturbance, 22, 212
strategy to deal with computational delay, 265
study of saturation, 43
supervisory control and data acquisition (SCADA) system, 355
`testcmppc.m`, 232
`testexa.m`, 111, 116
`testsim.m`, 239
time-delay system, 144
tuning parameters, 105, 170
two degrees-of-freedom, 311
working constraint set, 60
xPC Target, 340, 350

Other titles published in this series (continued):

Soft Sensors for Monitoring and Control of Industrial Processes

Luigi Fortuna, Salvatore Graziani,
Alessandro Rizzo and Maria G. Xibilia

Adaptive Voltage Control in Power Systems

Giuseppe Fusco and Mario Russo

Advanced Control of Industrial Processes

Piotr Tatjewski

Process Control Performance Assessment

Andrzej W. Ordys, Damien Uduehi
and Michael A. Johnson (Eds.)

Modelling and Analysis of Hybrid Supervisory Systems

Emilia Villani, Paulo E. Miyagi
and Robert Valette

Process Control

Jie Bao and Peter L. Lee

Distributed Embedded Control Systems

Matjaž Colnarič, Domen Verber
and Wolfgang A. Halang

Precision Motion Control (2nd Ed.)

Tan Kok Kiong, Lee Tong Heng
and Huang Sunan

Optimal Control of Wind Energy Systems

Iulian Munteanu, Antoneta Iuliana Bratcu,
Nicolaos-Antonio Cutululis and Emil
Ceanga

*Identification of Continuous-time Models
from Sampled Data*

Hugues Garnier and Liuping Wang (Eds.)

Model-based Process Supervision

Arun K. Samantaray and Belkacem
Bouamama

*Diagnosis of Process Nonlinearities
and Valve Stiction*

M.A.A. Shoukat Choudhury, Sirish L.
Shah, and Nina F. Thornhill

Magnetic Control of Tokamak Plasmas

Marco Ariola and Alfredo Pironti

Real-time Iterative Learning Control

Jian-Xin Xu, Sanjib K. Panda
and Tong H. Lee

*Deadlock Resolution in Automated
Manufacturing Systems*

ZhiWu Li and MengChu Zhou

*Fault-tolerant Flight Control
and Guidance Systems*

Guillaume Ducard
Publication due May 2009

Design of Fault-tolerant Control Systems

Hassan Noura, Didier Theilliol,
Jean-Christophe Ponsart and Abbas
Chamseddine

Publication due June 2009

Predictive Functional Control

Jacques Richalet and Donal O'Donovan
Publication due June 2009

*Advanced Control and Supervision
of Mineral Processing Plants*

Daniel Sbárbaro and René del Villar (Eds.)
Publication due July 2009

*Stochastic Distribution Control
System Design*

Lei Guo and Hong Wang
Publication due August 2009

*Detection and Diagnosis of Stiction
in Control Loops*

Mohieddine Jelali and Biao Huang (Eds.)
Publication due October 2009

*Active Braking Control Design for Road
Vehicles*

Sergio M. Savaresi and Mara Tanelli
Publication due November 2009