TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY
Julkaisu 647 • Publication 647

Xianghua Wei

Application of a Semi-Analytical Method to Model Predictive Control

Tampere 2007

Tampereen teknillinen yliopisto. Julkaisu 647
Tampere University of Technology. Publication 647

Xianghua Wei

Application of a Semi-Analytical Method to Model Predictive Control

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in Konetalo Building, Auditorium K1702, at Tampere University of Technology, on the 19th of January 2007, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2007

ISBN 978-952-15-1703-7 (printed)
ISBN 978-952-15-1806-5 (PDF)
ISSN 1459-2045

Abstract

This thesis proposes a semi-analytical algorithm, named repetitive optimal open-loop control (ROC), based on the Model Predictive Control (MPC) framework to generate open-loop feedback control for solving dynamic nonlinear optimal control problems with constraints. The algorithm is developed for the continuous-time NMPC. The generated feedback law builds a semi-analytical solution between the optimal control variables and states. The resulting optimal control trajectory is well defined in a "continuously" varied sequence.

The optimal control problem is converted into a two-point boundary-value problem (TPBVP) form, and solved by a back-and-forth shooting method. State and output constraints are dealt with the penalty function approach. The Kalman filter is used for state estimation. Implementation of ROC algorithm is done: algorithm competency testing with a hydro-electric power plant chain experiment; and normal solution proposal for optimal control problem with an exothermic chemical reactor application. Results prove out without any doubt it is a promising optimal control algorithm for handling fairly complicated constrained nonlinear dynamic systems.

Keywords: model predictive control (MPC), repetitive optimal open-loop control (ROC), two-point boundary-value problem (TPBVP), back-and-forth shooting.

Acknowledgements

This thesis work has been carried out in the Automation and Control Institute of Tampere University of Technology during the years 2003 - 2006.

I wish to hereby express my sincerest gratitude to my supervisor Professor Pentti Lautala, the head of the Automation department, for his kind guidance and support for this work. His difficult questions have pushed this work forward when it stalled and his insight has allowed me to progress when it seemed impossible. He has been a constant source of encouragement, knowledge, inspiration and wisdom during the course of my research.

I would also like to thank Prof. Raimo Ylinen and Dr. Juhani Henttonen for their gracious pre-examination work on my dissertation. I have gained much from their conversations, questions, and suggestions. I also want to thank Mr. James Rowland for English proofreading the manuscript.

My thanks are further due to the entire personnel of the Automation and Control Institute for creating pleasant and friendly working conditions. During my graduate studies here, I was fortunate enough to interact with a number of people from whom I have benefited greatly. I gratefully acknowledge all of you who have offered your advise, assistance, encouragement, and most importantly, friendship.

Finally, I want to thank my parents, my brother and all my friends for their invaluable

support and encouragement during these years.

Tampere, December 2006

Xianghua Wei

Table of Contents

List of Figures

List of Abbreviations, Nomenclature and Symbols

Δ	Difference operator
$\hat{d}$	Estimated disturbance
$\hat{x}$	Estimated state
$\hat{y}$	Predicted ouput
Ω	Input constraint
ϕ	Terminal state penalty
ψ	Terminal region
$A(q^{-1})$	Matrix polynomial in q^{-1}
F	Weight matrix on terminal state
H	Hamiltonian function
H_c	Control horizon
H_p	Prediction horizon
i	Iteration index
J	Performance index
k	Time, discrete-time
L	Integrand of cost function
L_{est}	Estimator gain
p	Adjoint (or costate) variable, $p \in R^n$
Q	Weight matrix on output
q^{-1}	Backward shift operator

R	Weight matrix on control
S	Weight matrix on control movement
t	Time, continuous-time
t_0	Initial time
t_f	Terminal time
u	Input variable, $u \in R^{m_u}$
u^*	Optimal control trajectory
u^o	Open-loop optimal control trajectory
v	Measured disturbance variable, $v \in R^{m_v}$
w	Unmeasured disturbance variable or noise, $w \in R^{m_w}$
x	State variable, $x \in R^n$
x^*	Optimal state trajectory
y	Output variable, $y \in R^p$
y_{ref}	Output reference trajectory
y_{sp}	Set-point of output variable
ANN	Artificial Neural Networks
ARIMAX	Auto-Regressive Integrated Moving Average with eXogenous
ARX	Auto-Regressive with eXogenous
CV	Controlled Variables or outputs
DAE	Differential Algebraic Equation
DMC	Dynamic Matrix Control
DP	Dynamic Programming
EKF	Extended Kalman Filter
EPSAC	Extended Prediction Self-Adaptive Control
FIR	Finite Impulse Response
FSR	Finite Step Response
GPC	Generalized Predictive Control
HJB	Hamilton-Jacobi-Bellman
LMPC	Linear Model Predictive Control

LQR	Linear Quadratic Regulator
LTI	Linear Time Invariant
MAC	Model Algorithmic Control
MBPC	Model-Based Predictive Control
MD	Measured Disturbance
MHC	Moving Horizon Control
MHE	Moving Horizon Estimation
MIMO	Multi-Input Multi-Output
MPC	Model Predictive Control
MPHC	Model Predictive Heuristic Control
MV	Manipulated Variables or inputs
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
ODE	Ordinary Differential Equation
PFC	Predictive Function Control
QDMC	Quadratic Dynamic Matrix Control
QP	Quadratic Programming
ROC	Repetitive optimal Open-loop Control
SOLO	Sequential Open Loop Optimization
SQP	Sequential Quadratic Programming
TPBVP	Two-Point Boundary-Value Problem
UMD	Unmeasured Disturbance

Chapter 1

Introduction

This chapter provides a background and motivation of this research work, and the thesis's contributions as well as its content outlines.

1.1 Background and Motivation

Model Predictive Control (MPC), also known as a receding horizon control (RHC) or moving horizon control (MHC), originally developed to meet the specifications of control needs to power plants and oil refineries [QB96], has become nowadays an attractive feedback control strategy in a wide variety of application areas including chemicals, food processing, automotive, aerospace, metallurgy, and pulp and paper industry, etc [QB96]. MPC refers to a class of control algorithms which use an explicit dynamic process model to predict the plant's future response and optimize its performance. Its concept of using an open-loop optimal control computation to synthesize a feedback controller is so natural that it probably occurred to many researcher in the optimal control field during the last two decades [Mac02, MR93].

The ideas of MPC can be traced back to the 1960s when research on open-loop optimal control was a topic of significant interest [Mac02]. The idea of a receding horizon, which is the core of all MPC algorithms, was proposed by [Pro63]. Another early work related to MPC can be found in [LM67], however, the true birth of predictive control took place

at the end of the 1970s with the first publication from Richalet *et al.* [RRTP78] about model predictive heuristic control (MPHC). The main reasons for increasing acceptance of MPC technology by the process industry since 1985 are clear [CB99]:

- MPC is a model based controller, which can handle processes with long time-delays, non-minimum phase, unstable and nonlinear processes.
- It is an easy-to-tune control method, in principle, there are only few basic parameters to be tuned.
- Industrial processes have their limitations in valve capacity, technological saturation or requirements. MPC can handle such kinds of constraints in a systematic way during the design and implementation for the controller.

There are a number of names for representing particular states of predictive control, usually with corresponding product names and acronyms, such as:

- Model predictive heuristic control (MPHC)
- Dynamic matrix control (DMC)
- Extended prediction self-adaptive control (EPSAC)
- Generalized predictive control (GPC)
- Model algorithmic control (MAC)
- Predictive functional control (PFC)
- Quadratic dynamic matrix control (QDMC)
- Sequential open loop optimization (SOLO)

and so on. For more detailed information about those algorithms and their developments, see [CB99, Mac02, MR93, QB96]. Generic names which have become widely used to denote the whole area of predictive control are finally named as model predictive control (MPC) and model-based predictive control (MBPC).

The success of MPC properties are well developed in the linear case with constraints, see [MR93], the practices in industrial process control can refer to [RRTP78, QB96], but fewer results are available for the nonlinear case. However, the processes nowadays need to be operated under much tighter performance requirements to guarantee a safe but efficient environment than ever before. The question arises because the linear MPC (LMPC) is not suitable enough to satisfy the increasing requirements, and therefore a nonlinear MPC (NMPC) approach has been carried out. NMPC control allows the explicit consideration of a nonlinear process model with the state and input constraints. Many theoretical issues were published during the last twenty years, see [MM90, MM93, RMM93, Hen98, ABQ$^+$99, QB00, BDLS00, KM00, May00, DBS$^+$02, DFB$^+$04], etc. Although NMPC achieved some progression, the high on-line computation load still cannot be avoided, since at each sampling time the nonlinear optimal control problem has to be solved. There are only a few methods that address the reduction of "computational burden" from complex NMPC, see [BDLS99, DFB$^+$04]. Quite often some parts of problems have to be dealt with linearization [ÖKG00]. The interest in improving the efficiency and accuracy of NMPC control of nonlinear system and rapidly solving on-line optimization algorithm is the first reason for generating this work.

In addition, most of the MPC techniques have derived or focused on the discrete-time models, such as [RRTP78, CMT87a, CMT87b, RRM03]. The corresponding continuous-time approaches have still received relatively much less attention in the development [Lu95]. The question arises as to whether the similar strategies, such as prediction calculation, least-squares solutions which are applied in the discrete-time situation, can be easily adopted into continuous-time case. It is fair to say that, it is much more complicated and difficult to develop a continuous-time MPC in technique. The technical difficulty comes from: unlike the discrete MPC who is a finite dimensional optimization problem, the continuous-time MPC focuses on the infinite dimension which is always computationally more demanding than the finite dimension issue. The interest of extending the development of MPC in the continuous-time approach and overcoming the computational difficulty turn to be the second reason for generating this work.

1.2 Contributions of the thesis

The main contributions of this thesis are described as follows:

- Along the general lines of the MPC algorithm in discrete-time issues, this research has developed a derivation of the continuous-time approach.

- As we know, the discrete-time MPC is a finite dimensional decision optimization problem while continuous-time MPC is the infinite dimensional decision optimization problem. Normally, infinite decision needs heavier calculation than finite decision. ROC control, which is proposed by the research work, has contributed in handling nonlinear dynamics systems, and enhances the research extension of MPC field in the continuous area.

The ROC algorithm was implemented in the MATLAB environment. The application experiments with ROC control show quite good performance. The back-and-forth shooting method for solving the two-point boundary-value problem (TPBVP) is presented. Discussion about handling nonlinear dynamic system with state and input constraints using the ROC control optimization is also shown. And then studies of ROC control's capability of dealing with the zero-mean white Gaussian distributed noise in both linear and nonlinear dynamic systems using state estimation via Kalman filtering are also presented.

1.3 Contents of the thesis

The outline of this thesis is organized as follows:
Chapter 2 contains a description of the basic principle and algorithm of the standard MPC, which is the theoretical framework of the ROC method. At the beginning, a feedback structure of MPC is illustrated and its attractive features are specified. Then the most common components for building a MPC algorithm are introduced respectively. Later the chapter mainly focuses on presenting the development of optimization algorithm approaches based on MPC, and at the end of this chapter, the stability of MPC is

described.

Chapter 3 provides the central idea of this thesis work. A semi-analytical form optimal control method, based on the standard MPC scheme, is recommended for solving constrained dynamic nonlinear system, and calculating the future prediction in continuous-time form. The method generates an optimal open-loop feedback control idea for the controllable dynamic nonlinear system which allows the state and control variables to be constrained. The chapter starts with an introduction of the main approaches of optimal control theory built so far: the prescriptions of the principle of optimality, the minimum principle of Pontryagin, the derived Hamilton-Jacobi-Bellman (HJB) equation, and Bellman's dynamic programming (DP) equations are expatiated, respectively. Then an incorporation of optimal open-loop feedback control idea has been proposed, with a clear explanation of the ROC algorithm principle. Since this method is built on the basis of the standard MPC, the similarities and differences between them are compared (standard MPC is based on the MPC Toolbox, version MATLAB 7.04). At the end, some examples are introduced in for clearer illustration.

Chapter 4 reviews a numerical solution of the optimal control problem called back-and-forth shooting, an very effective algorithm that the ROC method relies on for solving special optimal control problem in this thesis. Algorithm convergence has been mentioned. And later two preliminary examples are given for illustration, which emphasizes that this algorithm is quite promising compared with traditional shooting algorithms for solving some special case.

Chapter 5 includes the application example's implementation. The chapter starts with an introduction of the first application: a hydro-electrical power plant chains, which has the most difficult form for solving the optimal control problem, a fixed end point issue. A continuous-time experimental mathematical model is built, a short-term optimization problem is formulated and reduced via Pontryagin's principle to a two-point boundary-value problem (TPBVP). The TPBVP is solved with the aid of the back-and-forth shoot-

ing algorithm. And the ROC controller is introduced for handling such continuous-time optimal control problem. The effect of various situations and computational performance are studied. The application study is meant to testify the capability of the ROC algorithm for dealing the difficult and complex optimal control problem and at the same time explicitly handling multiple constrained states and inputs very well. The second application is a multivariable nonlinear reactor with two inputs, two outputs and multiple steady states, which is a more industrial-like control problem well handled by the ROC algorithm.

Chapter 6 concludes the thesis with a summary and an outlook of interesting future improvement and development.

Chapter 2

Model Predictive Control

Model predictive control (MPC), is a control scheme which the ROC method relies on in this work, refers to a class of algorithms that calculate a sequence of manipulated variable adjustments in order to optimize the future behavior of a plant [QB96].

This chapter starts with a brief review of the MPC principle, general algorithm, a basic structure, and main features; in Section 2.2, the common components of the MPC setup are introduced respectively; later some different perspectives of optimization algorithms are studied in Section 2.3; and the stability of MPC is discussed in Section 2.4 as the end of this chapter.

2.1 Principles of MPC

In general, a MPC problem is formulated by solving numerically, on-line, at each sampling instant, in a finite horizon, a discrete-time optimal control problem subject to system dynamics and constraints of states and controls. A basic idea of the MPC scheme can be illustrated in Figure 2.1: To determine a predictive controller at time k by solving an optimal control problem over a prediction horizon $[k, k+H_p]$. The predicted output $\hat{y}(k+i|k)$ (for $i = 1 \dots H_p$) depend at the time k on the past input and output, and further on the assumed input trajectory $\hat{u}(k+i|k)$ (for $i = 0 \dots H_p - 1$), which will be applied over the

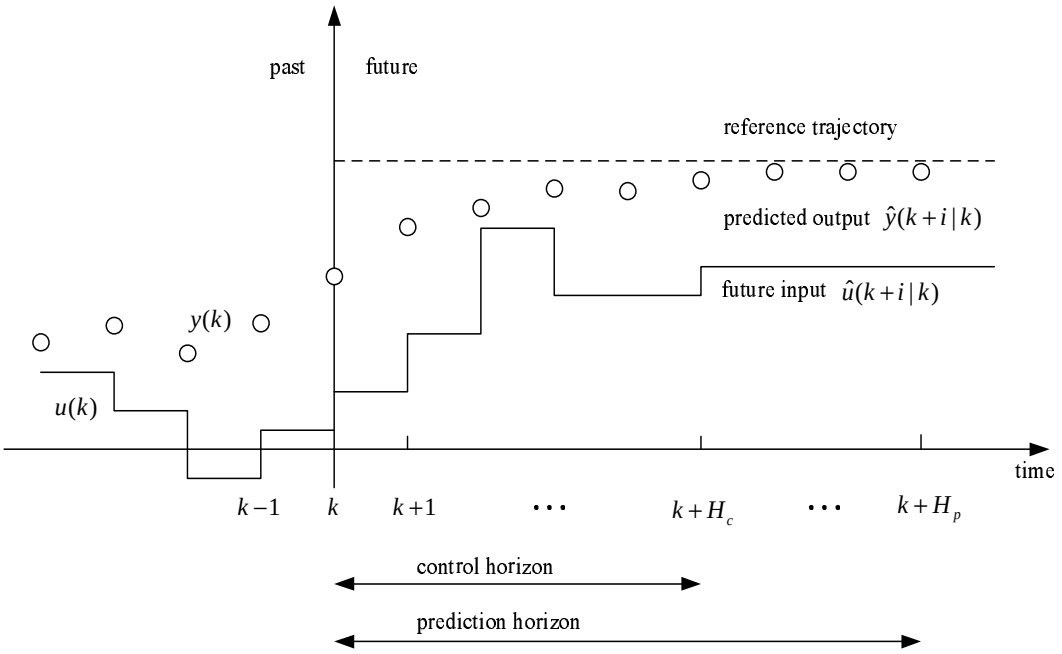


Figure 2.1: Basic principle of MPC strategy

prediction horizon [Mac02]. The path of future control variable movement is calculated by optimizing some criterion in order to keep the process as close as possible to the predefined reference trajectory. The criterion is commonly a form of quadratic function that consists of two parts: one part is to penalize the predicted output signal deviation from the reference trajectory, and the other part is to penalize the control variable movement in order to minimize the control effort [Mac02]. The control sequence is calculated along a certain control horizon H_c (normally $H_c \leq H_p$) to optimize a performance index, and the control input is assumed to vary within the control horizon H_c and remain constant thereafter.

In the end, the first control variable $\hat{u}(k|k)$ in the optimal trajectory is taken into the plant while the rest of the predicted control variable trajectories are discarded. The whole cycle of output measurement, prediction, and input trajectory determination is repeated one sampling interval forward [Mac02]. At next sampling instant $k+1$, a new system output $y(k+1)$ is obtained, prediction is made over the horizon $k+1+i$ (for $i = 1 \ldots H_p$), and a new input trajectory $\hat{u}(k+1+i|k+1)$ (for $i = 0 \ldots H_p - 1$) is applied. The entire procedure is repeated at subsequent control intervals in order to get an updated control

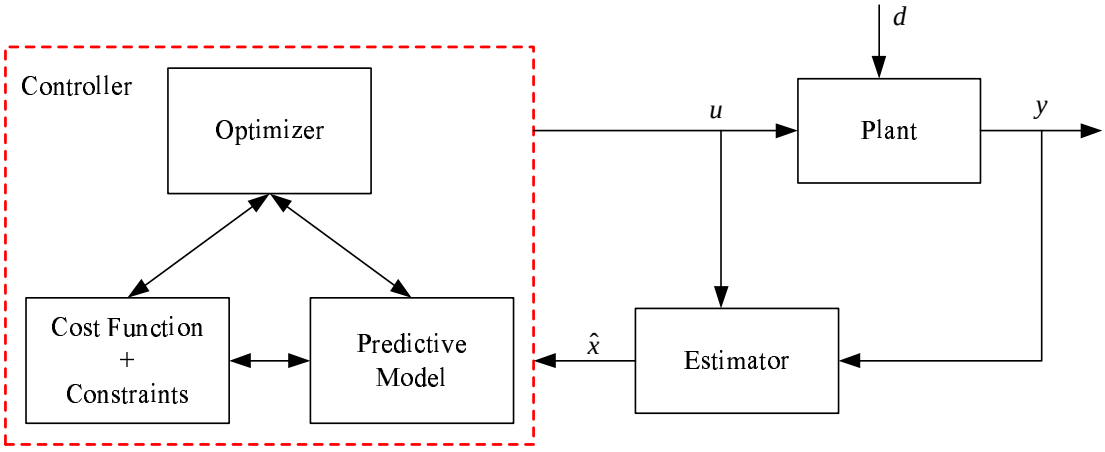


Figure 2.2: Basic structure of MPC

sequence, and horizons forward-move one time-step further.

A basic structure of MPC with feedback control loop is depicted by [ABQ+99, FA02] shown in Figure 2.2, where block *Plant* is the process to be controlled. Block *Controller*, which consists of predictive model, optimizer, and cost function with constraints, takes care of the solution of the open-loop optimal control problems over a prediction horizon. Block *Estimator* provides the controller with a feedback signal based on the measurements of inputs and outputs of the process. Each of the blocks may be nonlinear. The information provided by the feedback signal and how it is used to update the open-loop optimization depend on the specific choice of the controller and the estimator [ABQ+99, FA02].

Given some estimate of the process model, the entire algorithm starts to solve the open-loop optimal control problem, and the manipulated/control variable sequences are determined by minimizing some performance index over a prediction horizon. *Controller* is implemented in, and outputs of the process are then obtained. The new information got from the process is then used to update the open-loop optimization, and update is completed by estimating states or disturbances.

The main features of an ordinary MPC strategy are the following [ABQ+99, Mac02]:

[*1*] Explicit use of a model to predict the process output along a prediction horizon H_p, which in principle, allows the controller to handle process dynamics (e.g. dead times and lags) directly;

[*2*] Insertion a state estimator into feedback loop which provides better predictions and improves control performance;

[*3*] Consideration plant behavior (and the noise) over a future horizon and the formulation of the cost-function to be minimized, so that the effects of feed-forward and feedback disturbances can easily be anticipated and removed;

[*4*] Consideration of the process input, state and output constraints directly within the optimization formulation.

2.2 MPC Setup

To setup a general MPC algorithm, some common components, which are illustrated in Figure 2.2, have to be taken into account. These components are:

- Predictive model
- Optimization problem
- Constraints
- State and disturbance estimations

Hereafter, we will study these components respectively.

2.2.1 Predictive Model

MPC allows us to use the detailed knowledge of the process to construct a dynamic mathematical model. A good process model is very necessary for obtaining a higher performance control upon model-based algorithms. There are many different model forms used in MPC algorithms, only some of the most commonly used forms are listed below:

Finite Step Response (FSR) Model used by DMC [CB99]. For stable systems, the truncated response is given as: (we assumed that the sum is truncated and only N values are considered)

$$y(k) = \sum_{j=1}^{N} G_j \Delta u(k-j) = G(q^{-1})(1-q^{-1})u(k) \tag{2.1}$$

where $y(k) \in R^p$, $u(k) \in R^m$, and G_j is the jth element of step response of the process. Δ is the differencing operator $\Delta = 1 - q^{-1}$, q^{-1} is the backward shift operator $q^{-1}u(k) = u(k-1)$, and $G(q^{-1})$ includes the step response coefficient of the system:

$$G(q^{-1}) = G_1 q^{-1} + G_2 q^{-2} + \cdots + G_N q^{-N} \tag{2.2}$$

where $G_j \in R^{p \times m}$. Then the predictor will be:

$$\hat{y}(k+i|k) = \sum_{j=1}^{N} G_j \Delta u(k+i-j|k) \tag{2.3}$$

Noise is handled by the DMC algorithm in a way that the current modeling error, which is the difference between the predicted and measured output, is assumed to keep without changing over the whole prediction horizon [Hen96]. Step-like disturbances at system outputs are usually easy to handle because they enter into the controller in the same way as the set-point.

Finite Impulse Response (FIR) Model known as a weighting sequence model, it is used in MPHC [RRTP78]. The output is related to the input by:

$$y(k) = \sum_{j=1}^{N} H_j \Delta u(k-j) = H(q^{-1})(1-q^{-1})u(k) \tag{2.4}$$

where H_j is the jth element of impulse response of process. And the predictor is given:

$$\hat{y}(k+i|k) = \sum_{j=1}^{N} H_j \Delta u(k+i-j|t) \tag{2.5}$$

The FIR model is related to FSR model when we treat it as the difference between two steps with a lag of one sampling period through $H_j = G_j - G_{j-1}$. FSR or FIR model form is very intuitive because model parameters can be obtained from simple step response experiments, which avoid the selection of model order and identification of dead time [Ros03]. However, FSR and FIR models are not suitable if the system is unstable. When the system has slow modes for convergence, the truncation order can be quite high [Hen96].

Transfer Function Model Used in GPC [CB99] with the concept of transfer function $G = B/A$, the plant can be described by a input-output difference equation as:

$$A(q^{-1})y(k) = B(q^{-1})u(k) \tag{2.6}$$

where A, B are the polynomials given as:

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_a} q^{-n_a} \tag{2.7}$$

$$B(q^{-1}) = 1 + b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b} \tag{2.8}$$

where n_a, n_b are the orders of $A(q^{-1})$ and $B(q^{-1})$. So the predictor will be given:

$$\hat{y}(k+i|k) = \frac{B(q^{-1})}{A(q^{-1})} u(k+i|k) \tag{2.9}$$

This representation can be used for both stable and unstable process. Note that both FSR and FIR models can be seen as subsets of the transfer function model. This model requires less parameters than FSR or FIR, but prior knowledge of the process, especially the assumptions about the order of the A and B polynomials has to be made [CB99].

ARIMAX Model Described by an auto-regressive integrated moving average with exogenous inputs, ARIMAX model is used in GPC [CMT87a, CMT87b]. The output is given by:

$$A(q^{-1})\Delta y(k) = B(q^{-1})\Delta u(k) + C(q^{-1})e(k) \tag{2.10}$$

where A, B are same as before, and C is a polynomial like

$$C(q^{-1}) = 1 + c_1 q^{-1} + c_2 q^{-2} + \cdots + c_{nc} q^{-nc} \tag{2.11}$$

with the order of nc. $e(k)$ is assumed to be white noise with zero mean and variance σ_e. Thus the prediction is given:

$$\hat{y}(k+i|k) = \frac{B(q^{-1})}{A(q^{-1})}\Delta u(k+i|k) + \frac{C(q^{-1})}{A(q^{-1})\Delta}e(k) \tag{2.12}$$

The term Δ ensures integral action in the controller. The controller is therefore able to remove offset caused by step-like disturbance at the system output. The ARIMAX model can also be written in a state-space form.

State Space Model Used in RHTC [KB89], a general discrete-time linear time-invariant (LTI) state space model has been presented [CB99]:

$$\begin{aligned} x(k+1) &= Ax(k) + B_u u(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2.13}$$

where $x(k) \in R^n$ is a vector of n state variables, $u(k) \in R^{m_u}$ is a vector of m_u process inputs or manipulated variables, and $y(k) \in R^p$ is a vector of p process outputs or controlled variables. Then prediction for this model is given:

$$\hat{y}(k+i|k) = C\left[A^i\hat{x}(k|k) + \sum_{j=1}^{i} A^{j-1}B_u u(k+i-j|k)\right] \tag{2.14}$$

One of the advantages of the state-space representation is that it simplifies the prediction, however, system identification is more complex for a state-space model than for a transfer function model [CB99].

Nonlinear Model Normally linear models are used to be considered despite a fact that all industrial processes exhibit some degree of nonlinear behavior. This is due to the significant increase in complexity of the predictive control problem resulting from

the use of a nonlinear model. LMPC employs models which are linearized about the operating point to predict the response of the controlled process. This strategy proved to be quite successful even in controlling some moderate nonlinear processes [QB96, QB00]. However, the higher the degree of nonlinearity, the greater the level of mismatch between the actual process and the predictive model, thus resulting in the direct deterioration of the controller performance. Many industrial processes with high nonlinearities, have to ask for a more accurate description of the system dynamics and more precise control of process behaviors, the nonlinear model is built up in order to meet those specific demands [Hen98, QB00].

Nonlinear model, in general, is either based on fundamental principles or empirical observations or in the hybrid case with a mixture of both. The fundamental model is actually based on the conservation principles of mass, momentum and energy. Its advantage, well concluded by [Hen98] is that, as long as the underlying assumptions remain valid, the fundamental model can be expected to extrapolate to new operating regions where no data sets are available; however its drawback, pointed out by [Hen98] as well, is that the resulting dynamic model may be too complex and computationally time-consuming. The empirical model built from available process data may be more convenient in some cases since a detailed process understanding is not required for the model design, but a suitable model structure is still needed [Hen98].

Some of the recently published nonlinear model types used in NMPC include: Hammerstein and Wiener models [FPM97], Volterra models [POD96], polynomial ARX [SA97], artificial neural network models [CBG90, WMM$^+$00, SM97] and fuzzy logic models [TS85], which will be reviewed below, respectively, in order to demonstrate their most outstanding characteristics.

**** Hammerstein and Wiener Model*** For extending LMPC to the control of nonlinear processes, a model is required that can represent the nonlinearities but possibly without the complications associated with general nonlinear models. In order to fulfill

this need, some model structures which contain a static nonlinearity in series with a linear dynamic system have been developed in [FPM97]. When the nonlinear element proceeds the linear block, it is called the *Hammerstein model* as shown in Figure 2.3:

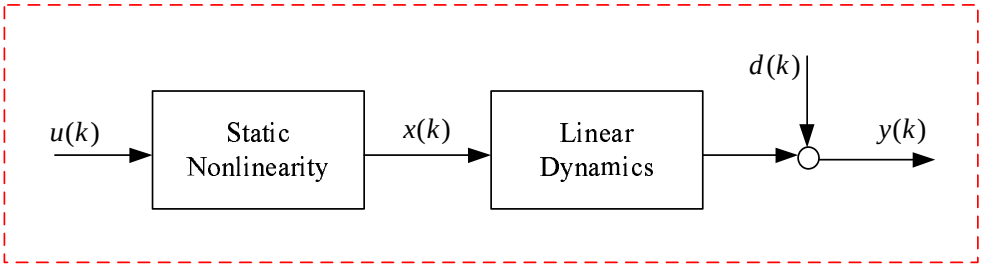


Figure 2.3: The Hammerstein model structure

Mathematically, the Hammerstein model is represented by the following equations:

$$y(k) = \frac{B(q^{-1})}{A(q^{-1})}x(k) + d(k) \tag{2.15}$$

where A, B are the polynomials given as:

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_a} q^{-n_a} \tag{2.16}$$

$$B(q^{-1}) = 1 + b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b} \tag{2.17}$$

the non-measured intermediate variable is given by:

$$x(k) = f(\theta, u(k)) \tag{2.18}$$

where q^{-1} is the unit delay operator, n_a, n_b are the orders of $A(q^{-1})$ and $B(q^{-1})$, $u(k)$ is the input, $y(k)$ is the output, $d(k)$ is the measured noise, $f(\cdot)$ is a static nonlinear function and θ is a set of parameters describing the nonlinearity.

If the linear and nonlinear blocks change the order, one obtains *Wiener model*, see figure 2.4. The Wiener model is represented by the following equations:

$$x(k) = \frac{B(q^{-1})}{A(q^{-1})}u(k) \tag{2.19}$$

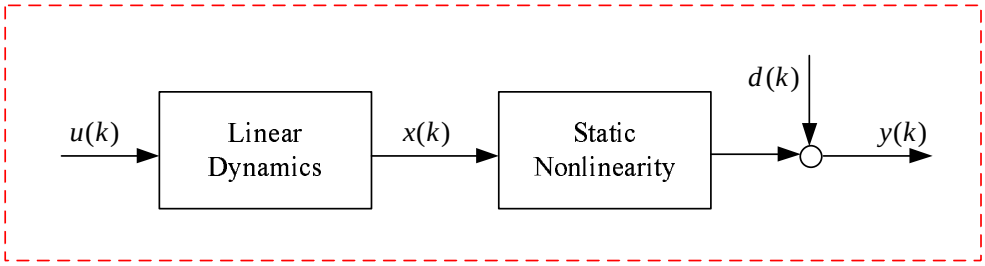


Figure 2.4: The Wiener model structure

and now the non-measured intermediate variable $x(t)$ is the input to the static nonlinearity given by:

$$y(k) = f(\theta, x(k)) + d(k) \tag{2.20}$$

Both the Hammerstein model and Wiener model consist of a process with linear dynamics but a nonlinear gain, and can represent many of the nonlinearities commonly encountered in industrial processes. Such models are particularly well suited for NMPC because LMPC is applied directly by transforming the input signal inverted into the static nonlinearity [FPM97].

**** Volterra Model*** Volterra models can be treated as natural extensions of linear FIR models to nonlinear FIR models by introducing cross products and polynomials of the inputs up to some order [SJ98]. This feature makes it particularly interesting for NMPC.

$$\begin{aligned} y(k) = y_0 + \sum_{j=0}^{\infty} a_j u(k-j) + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} b_{i,j} u(k-i)u(k-j) \\ + \sum_{l=0}^{\infty}\sum_{i=0}^{\infty}\sum_{j=0}^{\infty} c_{l,i,j} u(k-l)u(k-i)u(k-j) + \ldots \end{aligned} \tag{2.21}$$

However, the input prediction based on Volterra model keeps a nonlinearity, and therefore the NMPC optimization problem has to be solved by nonlinear programming (NLP). The NMPC algorithm based on second-order Volterra model has been applied in [DAOP95, MDOP96], with some comments about the model's performance as well. An obstacle of this model type to be an ideal choice for general nonlinear control problems is that

the large number of parameters explodes with the system's input dimension, therefore Volterra models beyond second order seem impractical [DAOP95].

**** Polynomial ARX Model*** Based on polynomial nonlinearities, a black-box identification of ARX model from designed experiments, is a fairly well developed technology [SJ98]. The nonlinear polynomial terms are determined by the structural identification, and the model based on a polynomial ARX model is given as

$$\begin{aligned} y(k) = y_0 &+ \sum_{j=1}^{n_\alpha} \alpha_j y(k-j) + \sum_{j=1}^{n_\beta} \beta_j u(k-j) \\ &+ \sum_{j=1}^{n_\alpha} \sum_{i=1}^{j} \rho_{ij} y(k-i)y(k-j) + \sum_{j=1}^{n_\beta} \sum_{i=1}^{j} \delta_{ij} u(k-i)u(k-j) \\ &+ \sum_{j=1}^{n_\alpha} \sum_{i=1}^{n_\beta} \eta_{ij} y(k-i)u(k-j) + \dots \end{aligned} \tag{2.22}$$

Although the NMPC optimization problem is non-quadratic and maybe non-convex in general, the polynomial structure of the model can be applied to compute the global optimum in some very special cases [SA97]. An experimental study of NMPC based on such a model, the waste water neutralization process, is described in [PK94].

Polynomial ARX model is improved than Volterra model in sense that the number of parameters needed for approximating a system in polynomial ARX model is generally much less than in the Volterra model because its previous output values can be used [SA97]. However, most process systems contain varying degrees of nonlinearity that may cause the accuracy reduction in this kind of model [SA97]. In order to overcome this accuracy loss shortcoming, many researchers in recent years have focused on implementing the neural networks as a tool of system identification.

**** Artificial Neural Networks (ANN) Model*** The most expensive part of the realization of a NMPC scheme is the derivation of an appropriate mathematical model. In many cases it is even impossible to obtain a suitable physically founded process model due to the complexity of the dynamic processes or the lack of knowledge about some

critical parameters of the models, such as temperature- and pressure-dependent mass transfer coefficients or viscosities, etc. To overcome these obstacles, an artificial neural network model is generated, just as a kind of nonlinear black-box model of the dynamic process, in order to modelling those unknown or poorly known systems. The ANN model can effectively model the complex characters of the nonlinear process and arbitrarily approximate nonlinear functions, particularly useful in dealing with a complex relationship between inputs and outputs.

In [LKB98], a concept of affine nonlinear predictors based on neural network is introduced so that predictive control algorithm is simple and easy to implement. They suggests a set of non-recursive predictors, which can compensate the influence of the time delay, to predict approximately the future output. These predictors use available sequences of past inputs and outputs of the process up to the sampling time to construct the predictive models, therefore the use of NLP technique for solving the nonlinear optimization problem is avoided.

[SA98] points out that the popular feed-forward neural networks show little robustness to disturbance, measurement noise, and changing operating regimes due to their high-order noise-sensitive input-output mapping. Therefore, they suggest the development of a reliable long-range predictor, comprising two neural networks with an external feedback in series. One network is used to predict the system state and output at the next sampling instant. The other network yields long-range predictions of the state and output, based on the state prediction from the first network. This modeling approach has the advantage that the prediction capability of the network model is improved by allowing efficient modeling of high-order input-output systems and incorporation of available analytical state-space models.

**** Fuzzy Logic Model*** Based on a smooth interpolation (fuzzy inference) between various pieces of data and models, Fuzzy model multiples local linear models valid in different operating regimes [SA97]. It provides a nonlinear mapping from input to output

with the capability of handling information presented in numerical or linguistic form. It can represent highly nonlinear processes and can smoothly integrate a priori knowledge with information obtained from process data. Therefore, an attempt to develop NMPC based on the fuzzy logic model has been tried by some researchers in recent years.

In [FJS95] a nonlinear fuzzy logic model based on interpolation of multiple local linear models is applied for on-line optimization using NLP technique. Some different approach is used in [FQ97] which suggests the interpolation of the solutions of multiple LMPC optimizations to approximate NMPC. Since it relies on multiple linear models, the fuzzy logic model is considered simpler than the ANN model. And in [JNS01] where an NMPC algorithm based on fuzzy logic model of Takagi-Sugeno type is taken and the fuzzy model interpolates between LTI models. The proposed NMPC strategy is based on linearizing the nonlinear product-sum fuzzy model around the current operating point, and compensate nonlinearity in process dynamics.

Regardless of the model forms and identification methods, the general nonlinear state-space model based on the form of previously introduced state-space model has also been taken into account frequently in NMPC field. In this thesis work, the state-space model is the mainly used model form for building the system processes.

2.2.2 Optimization Problem

Consider a general discrete-time nonlinear model based on the form of previously introduced state-space model:

$$\begin{aligned} x(k+1) &= f(x(k), u(k), v(k)) \\ y(k) &= g(x(k)) \end{aligned} \tag{2.23}$$

where f and g are some nonlinear functions, and $x(k) \in R^n$ is a vector of n state variables, $u(k) \in R^{m_u}$ is a vector of m_u process inputs or manipulated variables (MVs), $y(k) \in R^p$ is a vector of p process outputs or controlled variables (CVs), $v(k) \in R^{m_v}$ is a vector

of m_v measured disturbance variables (MDs). And then a prototypical expression of optimization problem for NMPC can be stated as [MR97]:

$$\min_{u(k|k),u(k+1|k),\ldots,u(k+H_c-1|k)} J(k) = \phi\left[\hat{y}(k+H_p|k)\right] + \sum_{i=0}^{H_p-1} L\left[\hat{y}(k+i|k), \hat{u}(k+i|k), \Delta\hat{u}(k+i|k)\right] \tag{2.24}$$

where ϕ and L are the nonlinear functions of their arguments. $\hat{y}(k+i|k)$ are the predicted controlled outputs at time k, $\hat{u}(k+i|k)$ are the predicted inputs which are calculated at time k, and $\hat{u}(k+H_c|k) = \hat{u}(k+H_c+1|k) = \ldots = \hat{u}(k+H_p-1|k)$. $\Delta\hat{u}(k+i|k) = \hat{u}(k+i|k) - \hat{u}(k+i-1|k)$ are the incremental values of the manipulated variables. The length of the prediction horizon is H_p and the control horizon is H_c.

The function of ϕ and L can be chosen to satisfy a wide variety of requirements, including minimization of the overall process cost. However, economic optimization may be achieved by a higher-level system which determines the appropriate set-point of the NMPC controller [Hen98]. Therefore, in general, it is meaningful to consider quadratic functions formed as follows:

$$\phi = \left[\hat{y}(k+H_p|k) - y_s(k)\right]^T Q \left[\hat{y}(k+H_p|k) - y_s(k)\right] \tag{2.25}$$

$$\begin{aligned} L = {} & \left[\hat{y}(k+i|k) - y_s(k)\right]^T Q \left[\hat{y}(k+i|k) - y_s(k))\right] \\ & + \left[\hat{u}(k+i|k) - u_s(k)\right]^T R \left[\hat{u}(k+i|k) - u_s(k))\right] \\ & + \Delta\hat{u}(k+i|k)^T S \Delta\hat{u}(k+i|k) \end{aligned} \tag{2.26}$$

where $u_s(k)$ and $y_s(k)$ are steady-state targets for u and y, respectively. $Q \geq 0$, $R > 0$ and $S > 0$ are the weights on tracking errors, controls, and control movements, and only the first H_c control movements are non-zero, $\Delta\hat{u}(k+i-1|k) = 0$ for all $i > H_c$ (assume $H_c \leq H_p$) [Mac02].

The output predictions are generated by setting inputs beyond the control horizon equal to the last computed values: $\hat{u}(k+i|k) = \hat{u}(k+H_c-1|k)$, for $H_c \leq i \leq H_p$. Note that the system model used to predict the future in the controller is initialized by the actual

system state, thus MPC requires measurements or estimates of the state variables, which will be discussed more in detail in Section 2.2.4.

2.2.3 Constraints

In practice, all processes are limited by some kind of constraints on inputs and outputs. The ability to explicitly handle constraints also makes MPC an attractive control method. Typical examples which have to consider input constraints are: a valve opening positions from fully open to fully close, the actuator limitations such as saturation and rate-of-change restrictions. Such input constraints have a general form

$$u_{min} \leq u(k) \leq u_{max} \tag{2.27}$$

$$\Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max} \tag{2.28}$$

where u_{min} and u_{max} are the minimum and maximum values of the inputs; and Δu_{min} and Δu_{max} are the minimum and maximum values of the rate-of-change of the inputs.

Outputs constraints are usually related to some operational limitations. Therefore, the consideration of safety and performance may let it turn to be necessary to set constraints on the system outputs. Normal examples like the levels in tanks, maximum pressure of boiler and temperature of chemical reactor, etc. Such constraints can be represented in a form as

$$y_{min} \leq y(k) \leq y_{max} \tag{2.29}$$

where y_{min} and y_{max} are the minimum and maximum values of the outputs.

State variable constraints (which is based on physical considerations) may also be specified if needed with a general form as

$$x_{min} \leq x(k) \leq x_{max} \tag{2.30}$$

By including constraints in the optimization problem, the controller can predict future

constraint violations and respond accordingly [QB96]. Solving the optimization problem with inequality constraints of inputs and outputs as the following:

$$u_{min} \leq \hat{u}(k+i|k) \leq u_{max}, \qquad 0 \leq i \leq H_c - 1 \tag{2.31}$$

$$\Delta u_{min} \leq \Delta\hat{u}(k+i|k) \leq \Delta u_{max}, \qquad 0 \leq i \leq H_c - 1 \tag{2.32}$$

$$y_{min} \leq \hat{y}(k+i|k) \leq y_{max}, \qquad 0 \leq i \leq H_p \tag{2.33}$$

Hard input and output constraints are handled within the general framework MPC algorithms based on the solution of a normal quadratic programming (QP) or a NLP problem [Hen98]. However, it is well known that hard output constraints can cause problems because the optimization may become infeasible and some of the constraints must then be relaxed or eliminated. And even in the case of a feasible solution the system may still become unstable due to active output constraints.

To cope with these problems and improve the constraint handling capabilities of MPC, [dOB94] introduces soft constraints by adding slack variables to the inequality constraints, and then the added the slack variables are treated as a penalty to the objective function to be minimized. In addition, [dOB94] also compares the use of quadratic and linear penalty formulations for dealing with hard constraints. The result shows that the use of a linear penalty formulation leads to the preferred stability properties compared to using quadratic penalty formulation with hard constraints in the optimization. Moreover, if the hard constrained controller is stable, the linear penalty formulation requires only finite penalty parameters to get the solution of the controller with hard output constraints. This characteristic allows better control of the errors resulting from constraint relaxation by using soft constraints.

2.2.4 State and Disturbance Estimation

The NMPC strategy follows the usual decomposition into: (a) an estimation problem, where states and (if desired) disturbances are estimated, and (b) a problem where the manipulated variables are computed by using the estimated states (and disturbances) as

the true initial states.

Estimation of states: MPC is an open-loop optimization strategy unless the state variables x are measurable [Mac02]. For many real systems, the state variable x cannot be measured explicitly. Therefore a standard approach in estimating the state of a dynamic system from input-output measurements is to use a state estimator to estimate the state variables, which is required to implement state feedback control strategies, see [Mac02]. The prediction of the future plant behavior is built upon the current available state variable values at time k. In practice, MPC techniques in particular require a precise knowledge of the state estimation of the system in order to solve the optimal control problem. The estimation of initial states is important for obtaining the correct estimates of the model parameters, especially in the nonlinear system, the output and the controller depend on the result of state estimation for good performance [RRM03], so if the state estimate is poor, both of them may fail.

The state estimator equation that is used for implementing to the discrete-time MPC is given by

$$\hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) + L_{est}\left[y(k) - C\hat{x}(k|k)\right] \tag{2.34}$$

where $\hat{x}(k+1|k)$ is the estimate of $x(k+1)$ and L_{est} is the estimator gain. $u(k)$ is determined from the optimal solution of the MPC scheme. The pair of (C, A) is detectable. In practice, the estimator gain L_{est} is often limited by the existence of measurement noise and has to be selected based upon the disturbance characteristics of the process [Mac02]. If statistical information is available about disturbances and measurement noise, then optimal estimate of state x can be obtained by using the Kalman filter (KF) [BH75, LR94, KS72, Mac02]. KF is an optimal estimator for unconstrained linear dynamic systems with Gaussian noise. It is popular due to its optimality and availability of a closed-form solution that makes estimation extremely efficient. The extended Kalman filter (EKF) is by far one of the most popular solution for on-line state estimation in nonlinear practical applications. It extends KF to nonlinear dynamic systems by linearizing the model at each time step, while assuming the noise and prior to be Gaus-

sian. State estimates are computed in nonlinear model and Kalman gain is calculated by linearizing the nonlinear system model. When a nonlinear continuous-time system model is defined as

$$\dot{x} = f(x, u) + n_1 \tag{2.35}$$

$$y = g(x) + n_2 \tag{2.36}$$

where n_1 and n_2 are random white Gaussian noises. With the initial system output measurement $y(k)$, the initial state estimate $\hat{x}(k|k-1)$, the input control variable $u(k-1)$, and the covariance matrices of process noise Q_n and R_n, an EKF algorithm for (2.35)-(2.36) can be given [Hen96, LR94] as:

[*1*] calculate the prediction error at time k

$$\hat{d}(k|k) = y(k) - g(\hat{x}(k|k-1)) \tag{2.37}$$

[*2*] linearize and discretize the nonlinear continuous-time system at the current control and previous state estimate and obtain the system matrices $A_d(k)$, $B_d(k)$ and $C_d(k)$;

[*3*] calculate the steady-state Kalman gain $K(k)$ with the linearized and discretized system matrices $A_d(k)$, $B_d(k)$, $C_d(k)$ and covariance matrices Q_n and R_n;

[*4*] update the state estimate by

$$\hat{x}(k+1|k) = f(\hat{x}(k|k-1), u(k-1)) + \hat{x}(k|k-1) + K(k)\hat{d}(k|k) \tag{2.38}$$

Since EKF extends the application of KF to estimate nonlinear dynamic systems by linearizing the nonlinear process model at each time step, it implements the same solution strategy as KF does, thus EKF also inherits all KF's merits. EKF is favorable mainly because of its simple algorithm and efficient computation characters, but it may diverge from the true state and cannot satisfy the process constraints [CBG05, Jaz70, May79].

When the disturbances of system input and outputs are step-like disturbances, which can be modelled like [Hen96]

$$\dot{z}_u = \xi_u \tag{2.39}$$

$$\dot{z}_y = \xi_y \tag{2.40}$$

where ξ_u is a zero-mean uncorrelated normally distributed random variable with the covariance matrix Q_{ξ_u}. Then the system model for the estimator is formed as [Hen96]

$$\dot{x} = f(x, u + z_u) + n_1 \tag{2.41}$$

$$\dot{z}_u = \xi_u \tag{2.42}$$

$$y = g(x) + z_y + n_2 \tag{2.43}$$

$$\dot{z}_y = \xi_y \tag{2.44}$$

Generally, the combined number of components in the vectors z_u and z_y cannot be greater than the number of system outputs since the states z_u and z_y must be observable. The states of z_u and z_y are uncontrollable [Hen96].

Moving horizon estimation (MHE) approach to on-line state estimation is an extension of the least-square batch estimation algorithm [RLR96]. Its properties have been widely studied, see [RLR96, Rao00, TR02, RRM03, LKHB05] and it has been shown to outperform extended Kalman filtering (EKF) by avoiding divergence and constraint violation. A general formulation of the moving horizon estimator was presented and an algorithm was developed with a fixed-size estimation window and constraints on states, disturbances, and measurement noise, see [Rao00, TR02]. MHE based on NMPC model is formed with a form [LKHB05]

$$x(k+1) = f(x(k), u(k), d(k)) \tag{2.45}$$

$$y(k) = h(x(k), u(k), d(k)) \tag{2.46}$$

Note that it is not restricted to this type form of models. Here $x(k)$ represent the states

of the model, $u(k)$ are the inputs or MVs, $d(k)$ are the disturbances and $y(k)$ are the measured outputs or CVs.

The basic strategy of moving horizon estimation (MHE) is to estimate the state vector based on a finite number of past measurement samples. The oldest measurement sample is discarded when a new sample becomes available [Rao00, RRM03]. Assume that an optimal estimate for the states $x(k+1)$ at time $k+1$ is desired. From (2.45) the optimal estimate denoted as $\hat{x}(k+1|k)$ can be obtained by

$$\hat{x}(k+1|k) = f(\hat{x}(k|k), u(k), \hat{d}(k|k)) \tag{2.47}$$

if the optimal estimates for $\hat{x}(k|k)$ and $\hat{d}(k|k)$ are given. Following the same way of thought, an optimal estimate for $\hat{x}(k|k)$ can be obtained then by

$$\hat{x}(k|k-1) = f(\hat{x}(k-1|k-1), u(k-1), \hat{d}(k-1|k-1)) \tag{2.48}$$

if the optimal estimates for $\hat{x}(k-1|k-1)$ and $\hat{d}(k-1|k-1)$ are given. Repeating this way of thought H_m times, then the optimal estimate $\hat{x}(k+1|k)$ can be obtained from integrating those state equations when $\hat{x}(k-H_m|k-H_m)$ and $\left\{\hat{d}(k-H_m|k-H_m), \ldots, \hat{d}(k|k)\right\}$ are known [LKHB05]. To obtain these latter estimates, we minimize a criterion that is a function of the difference between the measured outputs $y(k-i)$ and the predicted outputs

$$\hat{y}(k-i|k) = h(\hat{x}(k-i|k), u(k-i), \hat{d}(k-i|k)) \tag{2.49}$$

at the time instants $i = 0, \ldots, H_m$. Choosing an least-square formulations, the following dynamic optimization problem derived for MHE has to be solved [Rao00, LKHB05]:

$$\begin{aligned} \min_{\hat{x}(k-H_m|k-H_m), \left\{\hat{d}(k-H_m|k-H_m), \ldots, \hat{d}(k|k)\right\}} \sum_{i=0}^{H_m} \Big(& \|y(k-i|k) - \hat{y}(k-i|k)\|_Q^2 \\ & + \left\|\hat{x}(k+i|k) - f(\hat{x}(k|k), u(k), \hat{d}(k|k))\right\|_R^2 \Big) \end{aligned} \tag{2.50}$$

After obtaining the estimates $\hat{x}(k-H_m|k-H_m)$ and $\left\{\hat{d}(k-H_m|k-H_m), \ldots, \hat{d}(k|k)\right\}$,

an estimate of $\hat{x}(k+1|k)$ can then be obtained via integrating the model equations. For obvious computational reasons the time horizon H_m cannot be chosen arbitrarily large. Instead, a moving horizon formation is taken where the dynamic optimization problem is solved repeatedly at every sampling instant for a fixed time horizon H_m [LKHB05].

A very important characteristic of the MHE approach in contrast to EKF is that the constraints in the estimation of linear and nonlinear dynamic systems can be involved in, see [Rao00, RRM03, HR05]. For example, [RRM03] investigates MHE as an on-line optimization strategy for estimating the states of a constrained discrete-time system. The estimated states are determined on-line by solving a finite horizon state estimation problem. When the new solutions become available, the old solutions are discarded from the estimation window, and the finite horizon state estimation problem is resolved to determine the new estimate of the state. Results show the significant improvement in estimation performance by including constraints into the estimation. However, since MHE has to solve a constrained optimization problem over each moving window, a much heavier computational expense is required [HR05].

Estimation of disturbances: In practice, the mismatch between the actual measured and the predicted values of the controlled variables, which is often referred to as model mismatch, is regarded as the influence of output disturbances [Mac02]. The disturbance estimation problem is to predict future plant output over the prediction by using the estimated output disturbance from the difference between the actual and predicted output [Raw99], and the effects of the disturbance estimates is to shift the steady-state target of the regulator [MR97].

The simplest way is to generate the output targets $y_s(k)$ from the differences between the setpoints $y_{sp}(k)$ and the disturbance estimates [MR97]. In this method, the penalty on the inputs is eliminated ($R = 0$), so that the quadratic function L becomes:

$$L = [\hat{y}(k+i|k) - y_s(k)]^T Q [\hat{y}(k+i|k) - y_s(k))] + \Delta\hat{u}(k+i|k)^T S \Delta\hat{u}(k+i|k) \quad (2.51)$$

the output targets are calculated as follows

$$y_s(k) = y_{sp}(k) - \hat{d}(k) \tag{2.52}$$

$$\hat{d}(k) = y(k) - \hat{y}(k|k) \tag{2.53}$$

where $y_{sp}(k)$ are setpoints for the output variables, $y(k)$ are the actual measured outputs, $\hat{y}(k|k)$ are the predicted outputs, and $\hat{d}(k)$ are the estimated disturbances. This disturbance model assumes plant/model mismatch caused by a step disturbance in the output and the disturbance remains constant over the prediction horizon. Although these assumptions rarely hold in practice, the disturbance model really eliminates offset for asymptotically constant setpoints under most conditions [MR97].

2.3 Optimization Algorithms in MPC

For the case of a LTI process model, with a quadratic cost function and without constraints, an analytical solution to the optimization problem can be established; and in case there are constraints presented, but those constraints are convex, then the quadratic optimization problem which we have to solve is convex and can still be solved easily [Mac02]. When a nonlinear system with non-convex constraints is considered, a non-convex optimization problem has to be solved iteratively at every sampling time, which is the most common ways in NMPC. Several different attempts for solving nonlinear optimization problems in MPC have been released during the last thirty years.

[GM86] uses a successive linearization method to linearize the nonlinear models at the current operating point. The future process behavior is predicted based on the linearized model, while the effect of past input movements is computed by its original nonlinear model. [LR94] develops the method by re-linearizing the process model iteratively in each control interval in order to improve the accuracy of the linear model.

A Newton-type NMPC algorithm, corresponding to a constrained Gauss-Newton method,

has been developed by [LB88, LB89, dOB95]. They proposed to linearize a nonlinear state-space model, and the nominal input trajectory (or reference trajectory) is determined by the computation of the input trajectory from the previous sampling time. A new input trajectory is computed by solving a quadratic program once over the prediction horizon, and the quadratic optimization problem is solved based on the linearized model. In [LB89], it was assumed that the states are available or measurable, thus the state estimation was not considered there. In addition, the optimization method used in [LB88, LB89] is a kind of sequential quadratic programming (SQP) strategy, because only the inputs appear directly in the optimization problem, however, its poor initial guess may lead the predicted trajectories far away from desired reference trajectories [dOB95]. This often causes a strong nonlinearity in NLP and poor convergence behavior, especially to those unstable systems [LB89].

Some researchers also proposed NMPC algorithm by using general nonlinear programming (NLP) techniques for solving the optimization problems, see [ER90, Beq91]. They use the nonlinear, continuous-time state-space models and solve model equation in some specific optimization, i.e., SQP algorithm is often considered in solving these kinds of NLP problems. The shortcoming is that these algorithms are always computationally heavier than the previous methods.

[BDLS99, BDLS00] propose a multiple shooting approach based on a simultaneous approach to solve NMPC problems, taking the numerical integration into account, in particular by suitable adaptations of the SQP algorithm. The prediction horizon is divided into a number of smaller subintervals, and integrates with each of them so that a continuous input trajectory is obtained as the solution. Because the continuity of the state trajectory from one interval to the other is enforced on the NLP level only, dealing with unstable and strongly nonlinear system models are possible.

We can notice that many optimization algorithm solutions for NMPC have been investigated lately, however, an analytical solution in NMPC approach is usually impossible to

find , and normally a numerical optimization method has to be taken instead.

2.4 Stability of MPC

MPC is an open-loop optimal control incorporated feedback policy via the receding horizon idea and the state (and disturbance) estimates. The stability is almost impossible to guarantee because of the very complicated feedback structure of MPC. Different possibilities to achieve closed-loop stability are proposed in many well written surveys, see [ABQ+99, dNMS00, MRRS00]. Here only some of the main approaches for MPC are presented and no detailed proofs are given.

To achieve closed-loop stability for MPC using a finite horizon length have been proposed in [KG88, MM90, MHER95]. Most of these approaches modify the MPC setup so that stability of the closed-loop can be guaranteed independently for the plant and performance specifications. This is usually achieved by adding suitable equality or inequality constraints, and suitable additional penalty terms to the cost function. These additional constraints are usually not motivated by physical restrictions or desired performance requirement but only used to enforce stability of the closed-loop, therefore named as stability constraint [May00, MRRS00].

Zero terminal equality constraint The simplest way to enforce stability with a finite prediction horizon is to add a *zero terminal equality constraint* at the end of the prediction horizon [KG88, MM90, FA02, MHER95], i.e. to add an equality constraint as

$$x(t + H_p) = 0 \tag{2.54}$$

The stability of the closed-loop can be achieved if the optimal control problem has a solution at $t = 0$. From *Bellman's Principle of Optimality*, we know that the feasibility at one sampling instance can also lead to feasibility at the following sampling instances and a decrease in the value function [Bel57]. The main advantages of the zero terminal constraint are: the concept is quite simple and the application is then straightforward.

However, the disadvantage is that the system must be brought to the origin in finite time, which may cause the problem if the short prediction horizon lengths is used [FA02].

Infinite horizon MPC The specific approach to achieve stability is to consider an *infinite horizon* ($H_p \to \infty$). Since the feasibility at one sampling time also implies feasibility and optimality at the next sampling time [Bel57], the input and state trajectories computed as the solution of the MPC optimization at each specific instance, are equal to the closed-loop system trajectories. The remaining part of the trajectories after one sampling instance are the optimal solution at the next sampling instance. This fact implies a closed-loop stability [FA02].

Terminal region constraint In order to overcome the use of a zero terminal constraint, some researchers suggest the use of a *terminal region constraint* [FA02, MM90, CA98]:

$$x(t + H_p) \in \psi \tag{2.55}$$

where H_p is the prediction horizon, and ψ is the terminal region. Or to add a *terminal penalty term* $\phi(x(t + H_p))$ into the cost function as:

$$\min J(x(t), u(\cdot)) = \min \left(\phi(x(t + H_p)) + \int_t^{t+H_p} L(s, x(s), u(s)) ds \right) \tag{2.56}$$

where $u(\cdot)$ on $[t, t + H_p]$. Note that the terminal penalty term is not a performance specification that can be chosen freely. Rather ϕ and terminal region ψ are determined off-line such that stability has to be enforced [CA98, FA02].

Quasi-infinite horizon NMPC For nonlinear systems, the *quasi-infinite horizon* approach of [CA98] as well as the approach of [dNMS98] use a terminal cost in the objective function as a key element to achieve stability. When $t \in [t, \infty)$, the cost function can be split up into two parts

$$\min J^{\infty}(x(t), u(\cdot)) = \min \left(\int_t^{t+H_p} L(s, x(s), u(s)) ds + \int_{t+H_p}^{\infty} L(s, x(s), u(s)) ds \right) \tag{2.57}$$

where $u(\cdot)$ on $[t, \infty)$. When the trajectories of the closed-loop system remain within some neighborhood of the origin (terminal region) for the time interval $[t + H_p, \infty)$, an upper approximation of the second term can be made. [CA98] suggests that to determine a terminal region ψ so that a local feedback law $u(t) = Kx(t)$ asymptotically stabilizes the nonlinear system and makes ψ positively invariant for the closed-loop; and also add an additional terminal inequality constraint $x(t + H_p) \in \psi$, so that the second term of equation can be upper bounded by the cost function caused by the application of the local controller $u(t) = Kx(t)$. The predicted state will not leave ψ after $t + H_p$ since $u(t) = Kx(t)$ gives ψ a positive invariant [CA98, FA02]. Furthermore, the feasibility at the next sampling instance is guaranteed to dismiss the first part of $u(t)$ and replace it by the open-loop input. Therefore, by using the local controller for $t \in [t + H_p, \infty)$ and $x(t + H_p) \in \psi$, we obtain:

$$\min J^{\infty}(x(t), u(\cdot)) \leq \min \left(\int_{t}^{t+H_p} L(s, x(s), u(s))ds + \int_{t+H_p}^{\infty} L(s, x(s), Kx(s))ds \right) \tag{2.58}$$

and by choosing the terminal region ψ and the terminal penalty term in order to leads

$$\int_{t+H_p}^{\infty} L(s, x(s), Kx(s))ds \leq \phi(x(t + H_p)) \tag{2.59}$$

Substituting (2.59) into (2.58), we obtain

$$\min J^{\infty} \leq \min J(x(t + H_p), u(\cdot))) \tag{2.60}$$

which implies that optimal value of the finite horizon problem bounds that of the corresponding infinite horizon problem. Therefore, the prediction horizon can be treated as extending *quasi* to infinity [CA98, FA02].

Chapter 3

Semi-analytical Optimization Method based on MPC Scheme

In this chapter, an efficient semi-analytical optimization method is proposed to generate an optimal open-loop feedback control law for controlling dynamic nonlinear systems. The method is based on the MPC framework and allows explicit dealing with constraints. The chapter is organized with four main sections. First, we introduce some main approaches of optimal control theory built so far, then the principle of optimality which is the key idea behind optimal feedback control is presented, after then, the minimum principle of Pontryagin, Bellman's dynamic programming (DP) equations, and Hamilton-Jacobi-Bellman (HJB) equation are specified respectively as well. In the following section, an optimal feedback control principle has been brought in. Based on those theoretical foundations presented, an efficient semi-analytical optimization algorithm named Repetitive optimal Open-loop Control (ROC) is finally carried out, and the similarities and differences between the standard MPC and ROC algorithm are compared. At the end, some numerical examples are presented for illustration.

3.1 Introduction to Optimal Control

Optimal control theory can be regarded as a generalization of the calculus of variation which was initiated in 1696, with Bernoulli's Brachystochrone problem [Bry96]. The

calculus of variation has, since then, been built up by numerous famous mathematicians. The most influential contributions, like those of Euler, Langrange, Legendre, Hamilton, Jacobi, and Weierstrass, etc can still be found in modern optimal control problems, which are identified by the names of their creators [SW97]. In the 1950s Bellman developed the concept of dynamic programming (DP) which can be used to solve discrete optimal control problems [Bel57]. Later Kalman solved a problem with linear dynamics and integral quadratic cost function, showing that the optimal control is a linear feedback [Kal60]. This valuable example of the optimal control problem has been later named as the Linear Quadratic Regulator (LQR). In 1956, Pontryagin *et al.* published the first necessary conditions of optimality for nonlinear optimal control problem, known as the Minimum (or Maximum) Principle [PBGM62]. These conditions are subsequently generalized in many ways for dealing optimal control problems with dynamic systems.

3.1.1 Optimal Control Problem

The optimal control problem considered here is defined in terms of a general nonlinear dynamic system modeled on the time interval $[t_0, t_f]$, and suppose that t_0 is the initial time, so that

$$\dot{x}(t) = f(t, x(t), u(t)) \tag{3.1}$$

with an initial time that

$$x(t_0) = x_0, \tag{3.2}$$

and

$$u(t) \in \Omega \tag{3.3}$$

where t_0 and t_f are fixed initial and terminal times of optimization, x_0 is a fixed state initial point and $x(t) \in R^n$, $t \in R$ and $u(t) \in R^m$. The state trajectory x is assumed to be a continuous function of time on $t \in [t_0, t_f]$, and input u is a bounded piecewise continuous function, which satisfies the constraints $u(t) \in \Omega$ for $t \in [t_0, t_f]$.

To solve this problem and guarantee the steady state in the end, we have to add the

penalty function for that, therefore a performance index is given as

$$J = \phi(x(t_f)) + \int_{t_0}^{t_f} L(t, x(t), u(t))dt \tag{3.4}$$

where ϕ is a function of the terminal state, L is a function of the state and control. Thus, the optimal control problem is to determine an admissible control $u(t) \in \Omega$ which minimizes the performance index above.

Note that the above mentioned optimal control problem is the most common and simplest type of such kinds of problems we can form and solve, but actually there are plenty of other initial setting formats, for more details see [PBGM62, AF66]. And the most difficult formation among is to solve a situation with the fixed state end point $x(t_f)$, i.e., the initial condition like

$$x(t_0) = x_0, \quad x(t_f) = x_f, \tag{3.5}$$

and its corresponding performance index looks like

$$J = \int_{t_0}^{t_f} L(t, x(t), u(t))dt \tag{3.6}$$

The proposed ROC algorithm will be used to testify its capability for handling the most difficult forms of optimal control problem with an application in the later chapter. However most optimal control problems in real industry can be described using free end points formulation, so the ROC algorithm will also show its capability to deal with those kinds of problems in later chapter as well.

3.1.2 Minimum Principle of Pontryagin

The minimum principle of Pontryagin, under smoothness hypotheses, gives the necessary conditions for optimal control problem [AF66]. The Hamiltonian function (or simply Hamiltonian) is defined as

$$H(t, x, u, p) = p^T f(t, x, u) + L(t, x, u) \tag{3.7}$$

where $x \in R^n$, $u \in R^m$ and adjoint (or costate) vector p satisfying $p \in R^n$. f is a function determining the system, L is the integrand of cost function, and $t \in [t_0, t_f]$. The optimal solution must be satisfy

$$\dot{x}^*(t) = \frac{\partial H}{\partial p}(t, x^*(t), u^*(t), p^*(t)), \tag{3.8}$$

$$\dot{p}^*(t) = -\frac{\partial H}{\partial x}(t, x^*(t), u^*(t), p^*(t)) \tag{3.9}$$

with the boundary conditions from initial condition (3.2)

$$x^*(t_0) = x_0, \quad p^*(t_f) = \left(\frac{\partial \phi}{\partial x(t_f)}\right)^T \tag{3.10}$$

Note that in this kind of optimal control problem the boundary condition has a special form. Because the initial state $x(t_0)$ should be known, the half of differential equations for state have fixed boundary conditions at the starting point. If the terminal state $x(t_f)$ is not fixed at the end, which is typical in most control problems, the other half of differential equations for adjoint state (or costate) $p(t_f)$ has to be fixed at the endpoint, this feature is utilized in this thesis. And if concerning the most difficult formation with the fixed state end point $x(t_f)$, then boundary conditions from initial condition (3.5) have to be satisfied instead

$$x^*(t_0) = x_0, \quad x^*(t_f) = x_f, \tag{3.11}$$

The function $H(t, x^*, u, p^*)$ has an absolute minimum as a function of u over Ω at $u = u^*$ that is

$$\min_{u \in \Omega} H(t, x^*, u, p^*) = H(t, x^*, u^*, p^*) \tag{3.12}$$

or equivalently,

$$H(t, x^*, u^*, p^*) \leq H(t, x^*, u, p^*), \quad \text{for all } u \in \Omega \tag{3.13}$$

Thus, the optimal problem we faced above is reduced to a two-point boundary-value problem (TPBVP), which can be solved with variety of numerical methods. Sometimes it is notoriously difficult to solve even with a high-speed computer. One of its numerical

solution algorithms will be well studied in next chapter.

3.1.3 Dynamic Programming

DP is a simple mathematical technique that has been used for many years by mathematicians, scientists and engineers in a variety of context. Bellman developed DP in conjunction with the appearance of digital computer into a systematic technique for optimal control theory. It is regarded as more universal than the calculus variations and the minimum principle of Pontryagin because of its power and simplicity based on the principle of optimality [Bel57], which states:

Theorem 3.1 (The Principle of Optimality) *An optimal control sequence has the property that, whatever the initial state and the optimal first control variable may be, the remaining controls (or decision variables) constitute an optimal control sequence with regard to the state resulting from the first control.*

The principle of optimality suggests that an optimal policy can be constructed in piecewise style that first to construct an optimal policy for the "subproblem" involving the last stage, and then extend the optimal policy to the "subproblem" involving the last two stages, and continue in this manner until an optimal policy for the entire problem is solved [BH75]. The DP algorithm is based on this principle so that it proceeds sequentially, by solving all subproblems in a given time length. So suppose that the system starts at $(t, x) : [t_0, t_f] \times R^n$ and proceeds for a short time Δt, then the optimal cost function becomes

$$J^*(t, x(t)) = \min_u \left(\int_t^{t+\Delta t} L(s, x(s), u(s))ds + J^*(t + \Delta t, x(t + \Delta t)) \right) \tag{3.14}$$

This is the DP equation which shows that the optimal control problem can be computed recursively backwards [Ber95]. By taking one more step, we derive a ***Hamilton-Jacobi-Bellman (HJB) equation***, which in general gives a more constructive way to determine the optimal cost function J^*. HJB equation is a central result in optimal control theory. Many other principles and design techniques follow from the HJB equation, which itself

is in fact just a statement of the DP principle in continuous-time with a general form as

$$-\frac{\partial J^*}{\partial t}(t,x) = \min_u \left(L(t,x,u) + \frac{\partial J^*}{\partial t}(t,x)^T f(t,x,u) \right) \tag{3.15}$$

where all $(t,x) \in [t_0, t_f] \times R^n$, and the boundary condition to HJB equation is

$$J^*(t_f, x) = \phi(x(t_f)) \tag{3.16}$$

In DP approach, when the optimal control $u^*(t)$ trajectory is got, the solution of all states $x(t)$ has to be found. An optimal state feedback law is then generated in a form of $u^*(t) = \tilde{u}(t, x, \frac{\partial J^*}{\partial t}(t,x))$ and can be valid for any initial condition, however a really "complete" solution under such state feedback law, in fact, is computationally very heavy to obtain and may cause the curse of dimension [Bel57, AF66]. In practice, only very low dimension system can be solved within a reasonable time.

3.2 Optimal Feedback Control

Iteration principle of optimality enables a recursive solution to the optimal control problem, known as optimal feedback control, which is based on the DP and HJB equations [Bel57, AF66]. The DP equation defines an optimal control problem in feedback or closed-loop form.

Reconsider the common form of optimal control problems from the nonlinear system (3.1) - (3.4). By solving a HJB partial differential equation, the optimal feedback control principle is implemented. Normally, the following steps are taken [AF66]:

[*1*] Define the Hamiltonian

$$H(t,x,u,p) = p^T f(t,x,u) + L(t,x,u) \tag{3.17}$$

where $p \in R^n$ is a adjoint (or costate) vector.

[2] Optimize pointwise over u to obtain

$$\begin{aligned}\tilde{u}(t,x,p) &= \min_{u\in\Omega} H(t,x,u,p) \\ &= \min_{u\in\Omega} \left\{p^T f(t,x,u) + L(t,x,u)\right\}\end{aligned} \tag{3.18}$$

[3] Solve the partial differential equation

$$-\frac{\partial J^*}{\partial t}(t,x) = H\left(t,x,\tilde{u}(t,x,\frac{\partial J^*}{\partial t}(t,x)),\frac{\partial J^*}{\partial t}(t,x)\right) \tag{3.19}$$

subject to the initial condition $J^*(t_f,x) = \phi(x(t_f))$.

Then the obtained $u^*(t) = \tilde{u}(t,x,\frac{\partial J}{\partial t}(t,x))$ is the optimal feedback control law.

We can clearly notice that the solution of HJB equation is usually pre-calculated for states at each time instant, thus an analytical solution, in general, is impossible to get (an exception is the Linear Quadratic Regulator (LQR)), and a numerical solution is also computationally very hard. In practice, only very low dimensional problems can be solved in reasonable time [Bel57, AF66], and HJB partial differential equation is normally difficult and may impossible to solve for the nonlinear system with inequality constraints [BH75].

3.3 Repetitive Optimal Open-loop Control

In previous sections, we have introduced two different approaches for solving the optimal control problem:

- Minimum principle of Pontryagin: a solution of the time-based piecewise constant optimal control is obtained, but it's only valid for the specified initial conditions. The solution of the optimal control problem can be reduced into a TPBVP problem, which sometimes is quite difficult to solve numerically even with a high-speed computer [FA02];

- DP and HJB partial differential equation: in order to get the optimal control trajectory, a solution for all states has to be found [Bel57]. The advantage is that it derives an optimal state feedback law which is valid for any initial condition; but the drawback is that to obtain a "complete" solution under such state feedback law, actually is computationally heavy and may cause the curse of dimensionality [Bel57], only very low dimension system can be solved in a reasonable time.

MPC algorithm uses an iterative on-line solution of open-loop optimal control problem. For LMPC, the solution of the optimal control problem can be converted into a solution of a convex, quadratic problem, which can be solved easily. This is one of the reasons why LMPC was widely used in many industry areas [MR93]. However, NMPC has to consider the nonlinear optimal control problem which is in general non-convex, and an analytical solution is very difficult to find, instead a numerical optimization solution has to be generated.

An idea comes up whether we can improve NMPC control by developing a new algorithm, which can hold the merits from both Minimum principle of Pontryagin and DP approaches but try to get an approximately analytical, or we say a semi-analytical solution to optimal control problems. Repetitive optimal open-loop control (ROC) is generated for such purpose by using an open-loop feedback laws to build a semi-analytical form relationship between the optimal control variables and states.

3.3.1 Optimal Trajectory Calculation for Dynamic Systems

Normally an optimal control problem can always be derived into a form of TPBVP via Pontryagin's minimun principle [PBGM62]. The relationship between control variables, states, and adjoint states is specified by the classical calculus of variation. Studies for this area have been extended to those problems containing inequality constraints on the control and state variables [RSR95]. And some efficient technical methods are introduced in [Hen96] for specifying the optimal control method into a TPBVP form by the aid of the MATLAB Symbolic Toolbox.

Consider the common optimal control problem of the Bolza form in [Hen96]

$$\min J = \min\Big(\phi(x(t_f), y_{ref}(t_f)) + \int_{t_0}^{t_f} L(x(t), u(t), y_{ref}(t))dt\Big) \tag{3.20}$$

where

$$\dot{x}(t) = f(x, u), \quad x(t_0) = x_0 \tag{3.21}$$

$$y(t) = g(x) \tag{3.22}$$

$$u(t) \in \Omega \tag{3.23}$$

where y_{ref} is the output reference trajectory, ϕ is a function of the terminal state, L is a function of the state and control, and both ϕ and L have continuous partial derivatives to x. Equation $f(x, u)$ and $g(x)$ are assumed to have continuous partial derivatives of their arguments. Input u is a bounded piecewise continuous function, which satisfies the constraints $u(t) \in \Omega$ for $t \in [t_0, t_f]$. t_0 is the fixed initial time of optimization, and t_f is the terminal time of optimization. The state trajectory x is supposed to be continuous on $t \in [t_0, t_f]$ [Hen96].

Function ϕ in (3.20) is introduced to ensure that the optimal solution will approach the reference trajectory y_{ref} in the end t_f. It has to be differentiable and a common form for the terminal output penalty with the terminal reference value from the reference trajectory $y_{ref}(t_f)$ is given as

$$\phi(x(t_f), y_{ref}(t_f)) = \big[g^T(x(t_f)) - y_{ref}^T(t_f)\big] Q \big[g(x(t_f)) - y_{ref}(t_f)\big] \tag{3.24}$$

where Q is a positive definite matrix. The reference trajectories in the numerical examples of this thesis are all set directly to constant initial setpoint values, i.e. $y_{ref}(t) = y_{ref}(t_0)$, where $t \in [t_0, t_f]$. Reducing (3.20) - (3.23) into a TPBVP via the minimum principle of Pontryagin. The L is formed as

$$L(x, u, y_{ref}) = \big[g(x) - y_{ref}\big]^T Q \big[g(x) - y_{ref}\big] + u^T R u \tag{3.25}$$

The predicted output is penalized with (3.24). The Hamiltonian function of the optimal problem is

$$H(x,u,p,y_{ref}) = \big[g(x) - y_{ref}\big]^T Q\big[g(x) - y_{ref}\big] + u^T R u + p^T f(x,u) \tag{3.26}$$

where $p \in R^n$ is the adjoint state. The optimal control u^* is obtained by minimizing the Hamiltonian function H. If the control variables are not constrained, the gradient of H with respect to u

$$\frac{\partial H}{\partial u} = (R + R^T)u + \Big(\frac{\partial f(x,u)}{\partial u}\Big)^T p = 0, \quad for\ u = u^* \tag{3.27}$$

However if control constraints exist, the optimal control u^* still has to minimize the Hamiltonian function H [Hen96]

$$H(x^*,p^*,u^*,y_{ref}) \leq H(x,p,u,y_{ref}), \quad for\ u^*,\ u \in \Omega \tag{3.28}$$

If the cost function is quadratic and control enters linearly into the system, then state x and adjoint state p in TPBVP has the form given by [Hen96]:

$$\dot{x} = f(x, u^o(x,p)) \tag{3.29}$$

$$\begin{aligned} \dot{p} &= -\frac{\partial H}{\partial x} \\ &= -\Big(\frac{\partial\big[g(x) - y_{ref}\big]}{\partial x}\Big)^T (Q + Q^T)\big[g(x) - y_{ref}\big] - \Big(\frac{\partial f(x,u^o(x,p))}{\partial x}\Big)^T p \end{aligned} \tag{3.30}$$

and the boundary conditions like:

$$x(t_0) = x_0, \tag{3.31}$$

$$\begin{aligned} p(t_f) &= \frac{\partial \phi(x(t_f), y_{ref}(t_f))}{\partial(x(t_f))} \\ &= \Big(\frac{\partial\big[g(x(t_f)) - y_{ref}(t_f)\big]}{\partial x}\Big)^T (Q + Q^T)\big[g(x(t_f)) - y_{ref}(t_f)\big] \end{aligned} \tag{3.32}$$

where $u^o(x,p)$ is derived from either (3.27) or (3.28).

However in true reality the control derivatives may have to be taken into account instead of control in optimal control problem

$$\min J = \min\Big(\phi(x(t_f), y_{ref}) + \int_{t_0}^{t_f} L(x(t), u(t), \dot{u}(t), y_{ref}(t))dt\Big) \tag{3.33}$$

where

$$\dot{x}(t) = f(x, u), \quad x(t_0) = x_0 \tag{3.34}$$

$$y(t) = g(x) \tag{3.35}$$

$$u(t) \in \Omega \tag{3.36}$$

where $\dot{u}(t)$ is control derivative. Reducing (3.33) - (3.36) into a TPBVP via the minimum principle of Pontryagin. The L is formed as

$$L(x, u, \dot{u}, y_{ref}) = \big[g(x) - y_{ref}\big]^T Q\big[g(x) - y_{ref}\big] + u^T R u + \dot{u}^T S \dot{u} \tag{3.37}$$

where the weight of control R is zero. Since the Hamiltonian function cannot have the derivatives, when $\dot{u}^T S_{roc} \dot{u}$ is needed to be taken into cost function, then the system model has to be extended to

$$\begin{bmatrix} \dot{x}(t) \\ \dot{u}(t) \end{bmatrix} = f\big([x\ u], v\big) \tag{3.38}$$

$$y(t) = g\big([x\ u]\big) \tag{3.39}$$

where the new state is $x_{new} = [x\ u]^T$ and input is $v = [0\ I]^T$. And Hamiltonian function is built up like

$$H(x_{new}, v, p, y_{ref}) = \big[g(x_{new}) - y_{ref}\big]^T Q\big[g(x_{new}) - y_{ref}\big] + v^T S v + p^T f(x_{new}, v) \tag{3.40}$$

Minimizing the Hamiltonian function H and the gradient of H with respect to v is

$$\frac{\partial H}{\partial v} = (S + S^T)v + \Big(\frac{\partial f(x_{new}, v)}{\partial v}\Big)^T p = 0, \tag{3.41}$$

If the cost function is quadratic and control enters linearly into the system, then new state x_{new} and adjoint state p in TPBVP has the form given:

$$\dot{x}_{new} = f(x_{new}, v(x_{new}, p)) \tag{3.42}$$

$$\begin{aligned} \dot{p} &= -\frac{\partial H}{\partial x_{new}} \\ &= -\Big(\frac{\partial \big[g(x_{new}) - y_{ref}\big]}{\partial x_{new}}\Big)^T (Q + Q^T)\big[g(x_{new}) - y_{ref}\big] \\ &\quad - \Big(\frac{\partial f(x_{new}, v^o(x_{new}, p))}{\partial x_{new}}\Big)^T p \end{aligned} \tag{3.43}$$

and the boundary conditions like:

$$x_{new}(t_0) = [x_0,\ u_0]^T \tag{3.44}$$

$$\begin{aligned} p(t_f) &= \frac{\partial \phi(x_{new}(t_f), y_{ref}(t_f))}{\partial (x_{new}(t_f))} \\ &= \Big(\frac{\partial \big[g(x_{new}(t_f)) - y_{ref}(t_f)\big]}{\partial x_{new}}\Big)^T (Q + Q^T)\big[g(x_{new}(t_f)) - y_{ref}(t_f)\big] \end{aligned} \tag{3.45}$$

where u_0 is an initial value of input u and the optimal control is got from $x_{new}(2, :)$.

Generally, when we have a linear optimization problem with a quadratic form cost function, the optimal control u^* can be solved in an analytical form paralleling with the TPBVP and a minimization of Hamiltonian function H. When the optimization problem is nonlinear but the cost function is still in quadratic form, if some fixed or distinguished local minimum can be still found because of min/max constraints, then the global minimum is guaranteed, and the optimal control u^* in some cases can be also solved in an analytical form. When the optimization problem is nonlinear and the cost function turns to be non-quadratic, without a guaranteed global minimum, then the optimal control u^* has to be solved numerically. In practice, when we have a one-dimensional optimization

problem, a numerical and linearized solution is possible to get with a reasonable computation time, however when the dimension is increasing, the computation is getting heavier and heavier and causes a dimension curse.

3.3.2 Introducing Feedback to Open-loop Optimization

Based on the principle of optimal feedback control, ROC drives the process inputs and outputs to their target values by introducing a feedback idea into the open-loop optimization manner. Feedback is inserted into ROC by solving repetitively the sequence of on-line open-loop optimal control problem Equations (3.20) - (3.23) with the aid of state (and disturbance) estimation and obtained optimal control trajectory is valid until next new system measurement becomes available. The whole procedure which includes both optimization and prediction is repeated to find a new input with the prediction horizons forward-moving.

Consider a sequence of sampling instants $\{t_i\}_{i\geq 0}$ with a sampling interval $t_s > 0$ (note that t_s is not necessary a constant value but definitely much smaller than the prediction horizon H_p), so that $t_{i+1} = t_i + t_s$ for all $i \geq 0$. The feedback control is obtained by repeatedly solving an open-loop optimal control problem at each sampling instant t_i, and the current state estimation $x(t_i)$ of the plant is used. Illustrated in Figure 3.1, the principle of ROC is specified in following steps, where x^o and u^o are the state trajectory and control solutions to an open-loop optimal control problem, and x^* and u^* are the optimal state trajectory and optimal control resulted from ROC strategy.:

[*1*] estimate or measure an initial state of the plant $x(t_i)$ at the current time point t_i (the first initial index $i = 0$);

[*2*] calculate the open-loop optimal control u^o at $[t_i, t_i + H_p) \rightarrow R^n$ (current initial index $i = 0$) and get the solution from TPBVP-form optimization problem;

[*3*] apply the open-loop optimal control $u^*(t) = u^o(t)$ immediately at the time interval $t \in [t_i, t_i + t_s)$ to the plant;

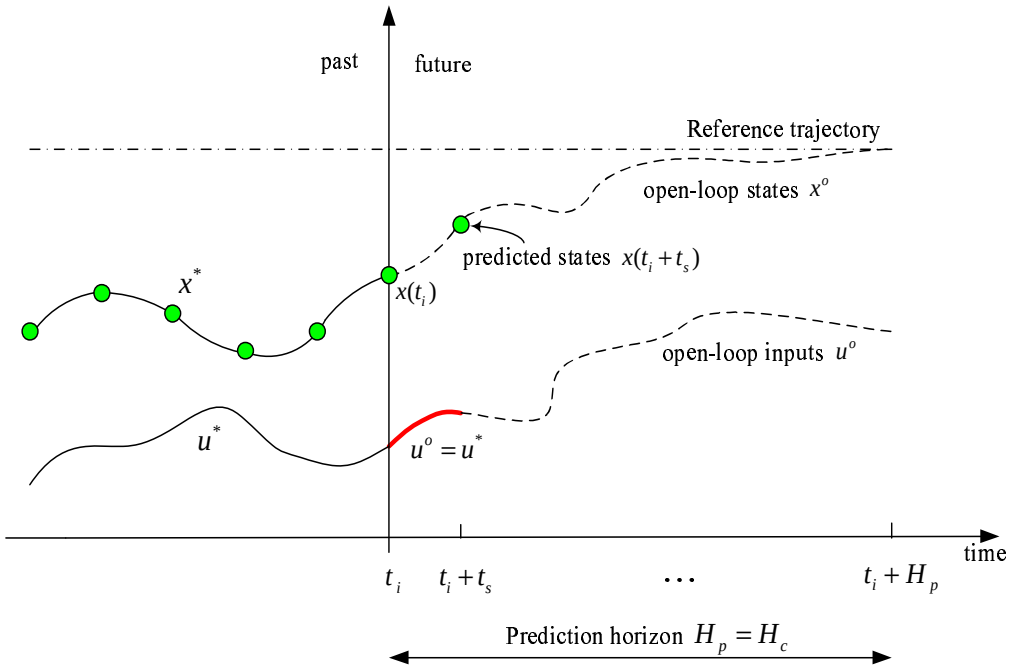


Figure 3.1: Principle of repetitive optimal open-loop control (ROC)

[4] repeat the procedure from step [1] for the next sampling interval t_{i+1} (the index i is incremented by one unit in each repetition), and move the prediction horizon forwardly one time-step further.

The optimal control $u^*(t)$, $t \in [t_i, t_i + t_s)$ is applied until the next sampling instant t_{i+1} is achieved and at that time a new optimal control problem is solved. Note although the control variables are stored on discrete-time intervals, the optimal control variables can be approximated at intermediate time values via for example ***spline interpolation*** [Hen96].

3.3.3 State Estimation based on ROC

Since it is normally impossible to measure the whole system states, an estimator is used for estimating the state variables of a dynamic system from input-output measurements based on the state feedback control strategies.

EKF Estimation based on ROC Extended Kalman filter (EKF) is used here for on-line state estimation in nonlinear practical applications. The EKF estimation is to

calculate the state estimates in nonlinear model while compute Kalman gain by a linearized discretized nonlinear continuous-time system model. At the initial time t_i (the first initial index $i = 0$), the system output measurement $y(t_i)$, the initial state estimate $\hat{x}(t_i|t_{i-1})$, the input control variable $u(t_{i-1})$, and the covariance matrices of process noise Q_{kal} and R_{kal}, the EKF estimation algorithm based on the ROC control can be given as:

[*1*] linearize and discretize the nonlinear continuous-time system at the control $u(t_{i-1})$ and previous state estimate and obtains the system matrices $A_d(t_i)$, $B_d(t_i)$ and $C_d(t_i)$;

[*2*] calculates the steady-state Kalman gain $K(t_i)$ with the linearized discrete system matrices $A_d(t_i)$, $B_d(t_i)$ and $C_d(t_i)$ and covariance matrices Q_{kal} and R_{kal};

[*3*] calculate the prediction error at time t_i

$$\hat{d}(t_i|t_i) = y(t_i) - g(\hat{x}(t_i|t_{i-1})) \tag{3.46}$$

[*4*] solve the optimal control variables from the open-loop optimization problem Equations(3.20) - (3.23) and (2.35)-(2.36). The initial state estimation solution is done by

a) the initial state estimate $\hat{x}(t_i|t_{i-1})$

b) assume that the prediction error is constant over the prediction horizon H_p

$$\hat{d}(t_i + j|t_i) = \hat{d}(t_i|t_i) = y(t_i) - g(\hat{x}(t_i|t_{i-1})), \quad j = 1, \ldots, H_p \tag{3.47}$$

c) solve the system over the prediction horizon

$$\hat{x}(t_i + j|t_{i-1}) = f(\hat{x}(t_{i-1} + j|t_{i-1}), u(t_{i-1} + j)) + \hat{x}(t_{i-1} + j|t_{i-1}) + K(t_i)\hat{d}(t_i|t_i) \tag{3.48}$$

The control variable values $u(t_i+j)$ are provided by the optimization algorithm. The output prediction is obtained from

$$y(t_i + j) = g(\hat{x}(t_i + j|t_{i-1}) + \hat{d}(t_i|t_i) \tag{3.49}$$

[5] apply to get new control variable $u(t_i)$

[6] estimate the next initial state $\hat{x}(t_{i+1}|t_i)$ by

$$\hat{x}(t_{i+1}|t_i) = f(\hat{x}(t_{i+1}|t_i), u(t_i)) + \hat{x}(t_{i+1}|t_i) + K(t_i)\hat{d}(t_i|t_i) \tag{3.50}$$

where the index i is incremented by one unit in each repetition.

Based on local linear approximations of state/measurement equations computed at each sample time, a recursive state estimator providing the minimum-variance state estimates, known as the extended Kalman filter (EKF) is derived.

MHE Estimation based on ROC Moving horizon estimation (MHE) approach to on-line state estimation is an extension of the least-square batch estimation algorithm [RLR96]. The state estimation problem is formed within a finite moving horizon window and to find the values of the unknown sequences (e.g. initial condition, state noise, measurement noise) in the least-square formulation [ML99]. When the unknowns are estimated, the states can be reconstructed by using the model. In the linear system with no constraints, the MHE is equivalent to the Kalman filter for choosing the certain weighting matrices [ML99]. Moving horizon estimation (MHE) is favored because: First, a nonlinear model can be used directly; at least within the estimation window, which improves the accuracy of estimation. Second, the constraints in the estimation of nonlinear dynamic systems is involved [ML99]. Adopted scheme proposed by Section 2.2.4 and minimizing the cost function

$$\min_{\hat{x}(t_i-H_m|t_i-H_m),\left\{\hat{d}(t_i-H_m|t_i-H_m),\ldots,\hat{d}(t_i|t_i)\right\}} \sum_{j=0}^{H_m} \Big(\|y(t_i-j|t_i) - \hat{y}(t_i-j|t_i)\|_Q^2 + \left\|\hat{x}(t_i+j|t_i) - f(\hat{x}(t_i|t_i), u(t_i), \hat{d}(t_i|t_i))\right\|_R^2 \Big) \tag{3.51}$$

After obtaining the estimates $\hat{x}(t_i - H_m|t_i - H_m)$ and $\left\{\hat{d}(t_i - H_m|t_i - H_m), \ldots, \hat{d}(t_i|t_i)\right\}$, an estimate of $\hat{x}(t_{i+1}|t_i)$ can then be obtained via integrating the model equations. The

time horizon H_m cannot be chosen arbitrarily large because of the heavier computational expense, so a fixed time horizon H_m is required.

Extended Kalman filter (EKF) is favored mainly because of its simplicity and efficient computation characters for handling the nonlinear dynamic systems, examples of implementation included estimation for the polymerization reactions, the production of silicon/germanium alloy films and the fermentation processes [HR05], however it may diverge from the true state and cannot satisfy the process constraints [CBG05, Jaz70, May79]. Moving horizon estimation (MHE) is famous because of its capability for handling the constraints in the estimation of nonlinear dynamic systems, examples of implementation included estimation for the chemical reaction systems, the batch reactor and CSTR [HR05], however because MHE has to solve a constrained optimization problem over each moving window, the heavy computational expense is obvious. In this thesis, the states from all experimental applications are assumed to be measurable. However when ROC control is implemented in some real industrial applications where normally all states cannot be measured, the EKF or MHE will be chosen depends on the exactly practical situation.

3.3.4 Comparison with General MPC

The general MPC mentioned here means the MPC Toolbox in MATLAB on the version 7.04. To compare the characteristics between these two predictive control approaches, the similarities are firstly considered to be emphasized:

- Both the ROC and the general MPC's framework are based on a relatively simple idea called receding horizon strategy as well as on their practical properties, namely the capability to cope with constraints and nonlinearities;
- Both of them use a dynamical model of the plant to predict the future behavior of the process based on the available data at the current time, and a future control signal is calculated so that the predicted output is as close as possible to the desired

reference trajectory;

- Both of them are carried out in an open-loop status, and this open-loop is "closed" by applying the feedback control law during each sampling interval, with a form of a function of states at some time instant.

- Both of them are formulated as the repeated solution of a finite horizon open-loop optimal control problem subject to system dynamics and input and state constraints.

The difference in the control performance between the ROC and the general MPC approaches is small when the constraints and disturbances acting on the plant to be controlled are not significant. Generally, most industrial systems are continuous-time form in nature, the advantage of ROC is that it allows flexibility to handle the continuous system with large variability in practice, and explicitly handle constraints and disturbances. Although the computational burden and complexity are arise, it is still quite applicable with the present day high-speed computation and powerful computer. Hereafter a few differences between them are specified respectively:

- In the ROC control approach, the model representing the dynamic process to be controlled, is exactly a nonlinear model while the general MPC prefers to use a linearized model from the original nonlinear dynamic process;

- ROC has a capability to explicitly deal with the continuous-time model while the general MPC only handles the discrete-time model, so if the continuous-time models are supplied in the general MPC, they are internally sampled with the controller's sampling time into discrete-time models;

- The performance index functions between them are different. Since the ROC is based on continuous-time form, the performance function is obtained with a continuous area, while the general MPC is based on discrete-time form, thus its performance index function is calculated as a connected stair-size field.

- In the ROC control strategy, the control horizon H_c is not used (or we say that the control horizon always equals to the prediction horizon, i.e. $H_c = H_p$); and actually in the general MPC scheme, the control horizon H_c has to be chosen carefully in order to maintain a good balance between the performance of control and the burden of computation: With the fixed prediction horizon, the smaller control horizon yields a more sluggish output response and more conservative input movements but computation won't be so heavy; a large control horizon has the opposite effect on performance [Hen98].

  Also, when the control horizon H_c is increased, more degrees of freedom have to be available for optimization, which is often translated into tighter control of the processes. Therefore, increasing the H_c may result in a better control system performance, but at the expense of larger changes in the manipulated variables and a reduction in the controller robustness, and an increase of heavier on-line computation [LH93];

- The control signal obtained from the ROC approach has not to be assumed as a form of piecewise constant between each sampling interval as the general MPC assumed. Instead, by calculating exactly according to the corresponding "approximately" continuous state (and the adjoint state) via *spline interpolation*, it has a form of piecewise varied sequence, which describes optimal control trajectory in continuous-time;

- Using zero-terminal equality constraint to achieve stability in the general MPC or ROC, which enforces the state at the end of the horizon to the desired steady-state, results the boundary condition for TPBVP reduced from the nonlinear optimal control problem. In practice, it is computationally expensive to be solved by the general MPC, but in ROC, a method called back-and-forth shooting is used to solve efficiently such TPBVP-formed optimization problem, thus the computation load is decreased.

Generally speaking, both ROC and general MPC are predictive control approaches based on the principle to solve iteratively on-line open-loop optimal control problem, however, they are quite different algorithms in the way of derivation, in the predictive model to be handled, in the cost function to be formulated and in the optimization freedoms to be concerned, etc.

3.4 Numerical Examples

In this section we would like make a comparison between ROC and general MPC more clearly by providing some illustrative numerical experiments. Note that the optimization problems considered are reduced into TPBVP-forms solved by the back-and-forth shooting method, which will be studied deeper in the next chapter. In addition, the optimal control problems concerned are converted into the solution of the convex, quadratic problems.

Example 1: Unconstrained Linear System In order to illustrate how ROC may be used to control system behavior, we firstly provide a very basic and simple example, an unconstrained linear system. The system is described with a transfer function as:

$$\frac{Y(s)}{U(s)} = \frac{1}{s(s+1)} \tag{3.52}$$

where the manipulated variable is $U(s)$ and the controlled variable is $Y(s)$. Then the discretized model for a sampling time of $T_s = 1.0$ second is given

$$\frac{Y(z)}{U(z)} = \frac{0.3679z + 0.2642}{z^2 - 1.368z + 0.3679} \tag{3.53}$$

The output reference trajectory is defined as constant 0.2 for the simulation. The initial time is set $t_0 = 0$ seconds and the end time is $t_f = 15$ seconds.

In ROC control, a prediction horizon is set to $H_p = 5$ seconds, and the control horizon is not used (or we say that the control horizon always equals to the prediction horizon in

ROC principle, i.e. $H_c = H_p$), and sampling time is $T_s = 1.0$ second. Since the system is integrating so the control trajectory will be trended to zero, so we could directly concern the weight of control instead of concerning the weight of control derivative here, and the weights of the output and the control are defined as

$$Q_{roc} = 10, \qquad R_{roc} = 0.01, \tag{3.54}$$

Note that the control weight in ROC cannot be zero otherwise the optimal control is infinite. The cost function in continuous-time is:

$$J_{roc} = \int_{t_0}^{t_f} \Big[(y - y_{ref})^T Q (y - y_{ref}) + u^T R u \Big] dt \tag{3.55}$$

In order to compare the result with ROC under the same condition, to general MPC control, the prediction horizon and control horizon are chosen the same as ROC, i.e., $H_p = 5$ seconds and $H_c = 5$ seconds, sampling time is also $T_s = 1.0$ second. The output weight and the control weight are defined as

$$Q_{mpc} = 10, \qquad R_{mpc} = 0, \tag{3.56}$$

From the comparison point of view, the control weight should be $R_{mpc} = 0.01$, the same value as R_{roc}. However as we know that when the weight of control is equal to zero, the system output can perform the fastest response in general MPC, therefore, to be fair to the general MPC, the weight of control is chosen $R_{mpc} = 0$. Then its cost function based on discrete-time is formed as:

$$J_{mpc} = \sum_{t_0}^{t_f} \Big[(y - y_{ref})^T Q (y - y_{ref}) + u^T R u \Big] \tag{3.57}$$

Applying ROC and general MPC to the system with both output tracking reference trajectories $y_{ref}(t) = 0.2$. Results are illustrated in Figure 3.2. It can be seen clearly that both approaches can solve the "nearly" equivalent problem, as we have to notify that there are still some differences between the problems precisely solved by ROC and gen-

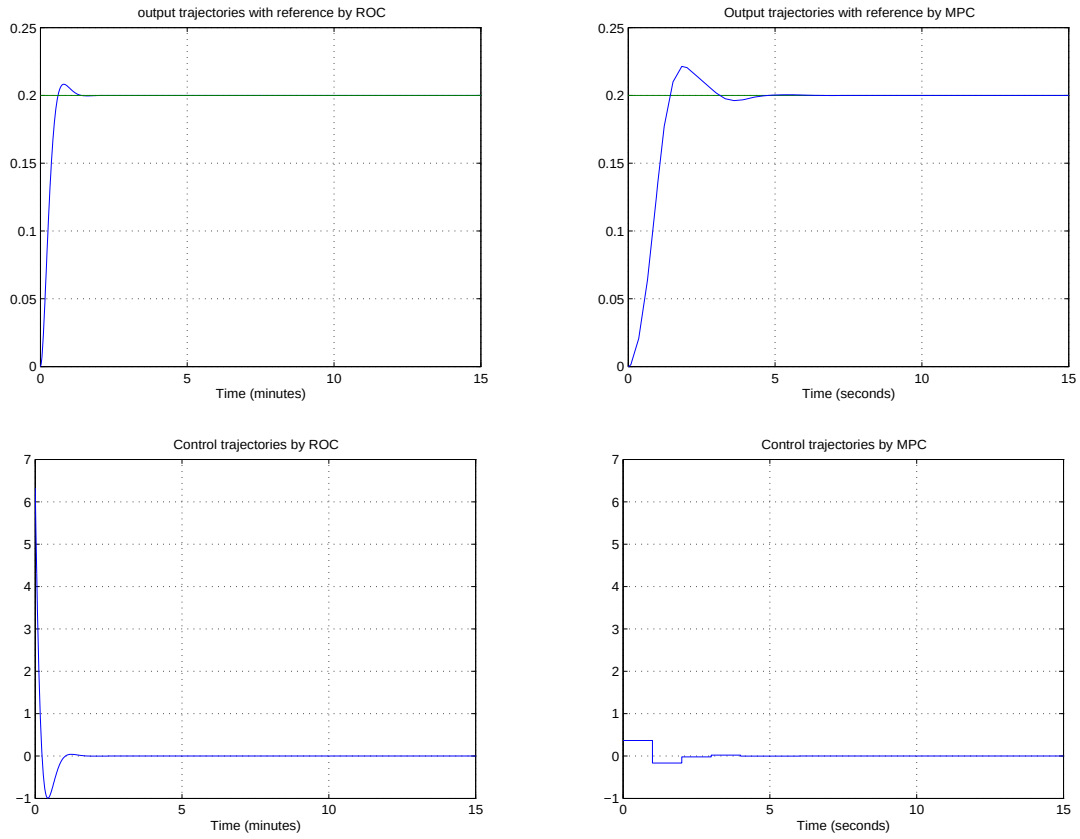


Figure 3.2: Unconstrained linear system controlled by ROC and general MPC

eral MPC: first, the differences between their performance index formulations, since one is built based on continuous-time and the other one is based on discrete-time, although by minimizing the stepsize in discrete-time can get very close cost function area as the continuous-time has, however they are not exactly the same; second, the freedom of the optimization is not equal, the ROC control horizon is defined in continuous-time and concerned with infinite variables, thus its freedom of the optimization is normally much higher than general MPC, whose control horizon is in discrete-time and concerned finite variables.

It can also be easily noticed that the start of the ROC optimal control trajectory is much higher than MPC control trajectory does because of the control weight. Although $R_{roc} = 0.01$ seems to be a quite small weight, however compared to $R_{mpc} = 0$, it is quite

a big value. Note that, hereafter, we will always compare these two control approaches based upon this principle. In addition, both variables, shown in the Figure 3.2, reach their set-points exactly while exhibiting some very small overshoot. Since without constraints and with linear dynamics, both algorithms perform nicely and efficiently. Hence the contributions of ROC and general MPC for unconstrained linear system are quite similar, but the system behaves smoother by ROC control than general MPC control in continuous-time.

As we know, in practice, the state variable x usually cannot be measured. The optimal control needs an initial state, which can be provided by a standard approach to estimate the state of a dynamic system from input-output measurements, and a Kalman filter K_{est} is used for estimating the state variables $\hat{x}$. Applying ROC principle with state estimation is as follows:

[*1*] estimate the state by $\hat{x}(k|k)$ for optimization using $y(k)$, $u(k-1)$ and $\hat{x}(k-1|k-1)$;

[*2*] calculate the optimal control over prediction horizon H_p;

[*3*] use the optimal control $u(k|k)$ in the plant;

[*4*] repeat the procedure from step [1] for the next sampling iteration.

Note in this case, although all data used are in discrete-time, the ROC control is still in a continuous-time form. Extending the state estimation to contain the white Gaussian distributed noise, the controller can cope with independent randomly with zero-mean white Gaussian distributed noise both at process input and output, etc. The covariance tuning parameters are defined as

$$Q_{kal} = 0.01, \quad R_{kal} = 0.01 \tag{3.58}$$

The initial state of the plant is $x_0 = [0,\ 0]$, and the estimator is $\hat{x}(k|k) = [0,\ 0]$. Then the optimal input trajectory and output trajectory turn to be as Figure 3.3, where the input disturbance is zero-mean white gaussian distributed noise with variance equals to

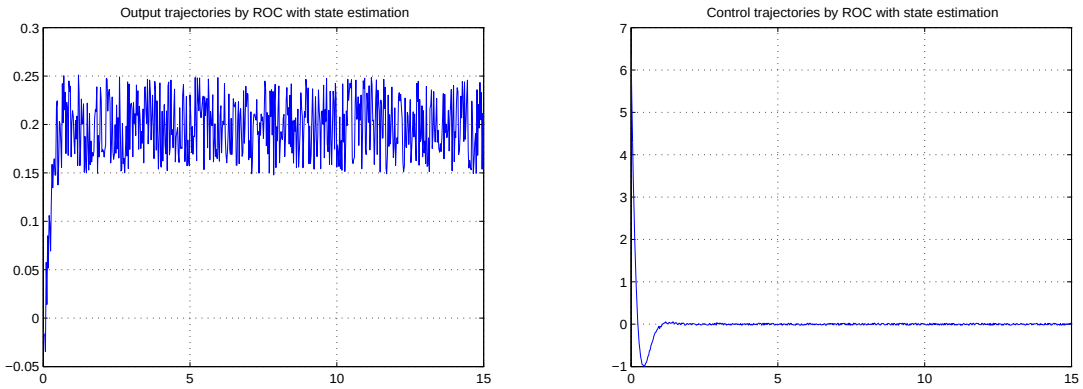


Figure 3.3: Unconstrained linear system controlled by ROC with state estimation

0.05, and output disturbance is zero-mean white gaussian distributed noise and variance equals to 0.1. Note that the pair (C, A) describing the overall state-space realization of the combination of plant and disturbance models has to be observable or detectable for a successful state estimation design. The model has been converted to discrete-time state-space form for obtaining the Kalman gain.

Example 2: Non-minimum Phase System In the second example, we consider a linear, stable non-minimum phase system given by the following transfer function:

$$G(s) = \frac{-s + 0.5}{s^2 + 5s + 6} \tag{3.59}$$

If the system is sampled at 1.0 second, the discrete transfer function is given by

$$G(z^{-1}) = \frac{-0.02775 + 0.09622z^{-1}}{1 - 0.1851z^{-1} + 0.00674z^{-1}} \tag{3.60}$$

The reference trajectory is defined as constant to $y_{ref} = 1.0$ for the simulation, and initial time is $t_0 = 0$ second, the end time is $t_f = 20$ seconds.

In ROC control, the prediction horizon (and control horizon) is chosen as $H_p = 10$ seconds (default $H_c = H_p$). When there is no integrator and the control derivative (or control movement to general MPC) won't turn to zero, we have to concern the weight of control

derivative instead of simple weight of control. The chosen output weight and the control derivative weight are as:

$$Q_{roc} = 10, \qquad S_{roc} = 0.01, \tag{3.61}$$

and the cost function forms in continuous-time is:

$$J_{roc} = \int_{t_0}^{t_f} \Big[(y - y_{ref})^T Q_{roc} (y - y_{ref}) + \dot{u}^T S_{roc} \dot{u} \Big] dt \tag{3.62}$$

with the state-space model $\dot{x} = Ax + Bu$. As Hamiltonian function cannot take the derivative, so when $\dot{u}^T S_{roc} \dot{u}$ is taken into cost function, then the system model has to be extended to

$$\begin{bmatrix} \dot{x} \\ \dot{u} \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} v \tag{3.63}$$

where the new state is $x_{new} = [x \ u]^T$ and input is v. And Hamiltonian function is

$$H = (C_{new} x_{new} - y_{ref})^T Q_{roc} (C_{new} x_{new} - y_{ref}) + v^T S_{roc} v + p^T (A_{new} x_{new} + B_{new} v) \tag{3.64}$$

where matrices A_{new}, B_{new}, and C_{new} are

$$A_{new} = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}, \quad B_{new} = \begin{bmatrix} 0 \\ I \end{bmatrix}, \quad C_{new} = \begin{bmatrix} C & 0 \end{bmatrix} \tag{3.65}$$

the derived TPBVP is written now like

$$\begin{aligned} \dot{x}_{new} &= A_{new} x_{new} - B_{new} (S_{roc} + S_{roc}^T)^{-1} B_{new}^T p \\ \dot{p} &= -\frac{\partial H}{\partial x} \\ &= -(C_{new}^T Q_{roc}^T C_{new} + C_{new}^T Q_{roc} C_{new}) x_{new} \\ &\quad + (C_{new}^T Q_{roc}^T + C_{new}^T Q_{roc}) * y_{ref} - A_{new}^T p \end{aligned} \tag{3.66}$$

In order to compare the result with ROC under the same condition, to general MPC control, the prediction horizon and control horizon are chosen the same as ROC, i.e.,

$H_p = 10$ seconds and $H_c = 10$ seconds, sampling time is also the same as ROC's, i.e. $T_s = 1.0$ second. The weighting factors applied to the system output and control movement are

$$Q_{mpc} = 10, \qquad S_{mpc} = 0.01, \tag{3.67}$$

and its corresponding cost function based on discrete-time forms as:

$$J_{mpc} = \sum_{t_0}^{t_f} \Big[(y - y_{ref})^T Q_{mpc} (y - y_{ref}) + \Delta u^T S_{mpc} \Delta u \Big] \tag{3.68}$$

In the first case, applying the ROC and the general MPC to the system with sampling time $T_s = 1.0$ second without output constraint, then their closed loop behaviors exhibit the typical non-minimum phase behavior with an initial peak in the opposite direction to the set-point change in Figure 3.4.

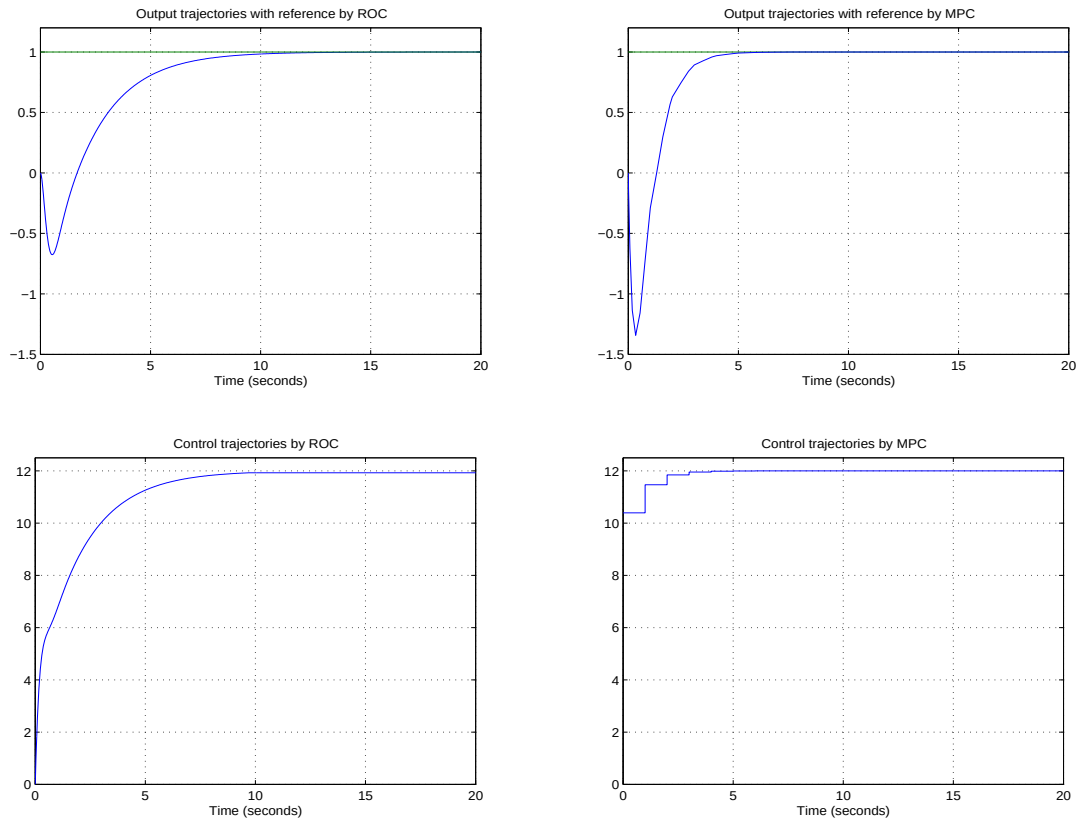


Figure 3.4: Non-minimum phase system controlled by ROC and general MPC

In addition, since the weight of the control derivative is taken into account here instead of a weight of control, the responses of output and control trajectories in the ROC control perform slower than in the general MPC, where a weight of the control movement is used instead.

As we know, the control of non-minimum phase systems remains an important problem in optimal control, one wishes to let the output of a dynamical system track a desired trajectory while trying to avoid inverse peak behavior, therefore the output constraint has to be concerned. If the response is preferred to obtain under a circumstance that the inverse peaks can be decreased, then a penalty function of output is required in ROC optimization control problem. The continuous-time cost function is built now as:

$$J_{roc} = \int_{t_0}^{t_f} \Big[(Cx - y_{ref})^T Q_{roc} (Cx - y_{ref}) + \dot{u}^T S_{roc} \dot{u} + s(2\frac{Cx}{y_{max}} - 1)^N \Big] dt \tag{3.69}$$

with the state-space model

$$\dot{x} = Ax + Bu \tag{3.70}$$

Since $\dot{u}^T S_{roc} \dot{u}$ is also taken into the cost function, the system model has to be extended as Equation (3.63) as well, where the new state is $x_{new} = [x \ u]^T$ and input is v. And Hamiltonian function is

$$\begin{aligned} H = &(C_{new}x_{new} - y_{ref})^T Q_{roc} (C_{new}x_{new} - y_{ref}) + v^T S_{roc} v \\ &+ s(2\frac{C_{new}x_{new}}{y_{max}} - 1)^N + p^T (A_{new}x_{new} + B_{new}v) \end{aligned} \tag{3.71}$$

where matrices A_{new}, B_{new}, and C_{new} are the same as (3.65). The derived TPBVP is written now as

$$\begin{aligned} \dot{x}_{new} &= A_{new}x_{new} - B_{new}(S_{roc} + S_{roc}^T)^{-1} B_{new}^T p \\ \dot{p} &= -\frac{\partial H}{\partial x} \\ &= -(C_{new}^T Q_{roc}^T C_{new} + C_{new}^T Q_{roc} C_{new})x_{new} + (C_{new}^T Q_{roc}^T + C_{new}^T Q_{roc}) * y_{ref} \\ &\quad - A_{new}^T p - \frac{2Ns}{y_{max}} C_{new}^T (2\frac{C_{new}x_{new}}{y_{max}} - 1)^{N-1} \end{aligned} \tag{3.72}$$

where s is a "small" positive constant and N is the penalty exponent with even integer values up to 12. In general, there are many possibilities to choose the form of the penalty terms. Note normally when outer penalty approach is used, the constraints are usually violated, because the small penalty may be at or near the border. The penalty grows only if the constraint is being violated. So if responses are preferred such no constraint violation happens, the tighter constraints have to be taken into account where in both ROC and general MPC optimal control problems, the minimum of output is chosen to be $y_{min} = -0.1$ and then give in ROC $y_{min_{roc}} = 0$ and in general MPC $y_{min_{mpc}} = -0.1$. Another approach is the inner penalty approach, but the results are about the same and shown in Figure 3.5.

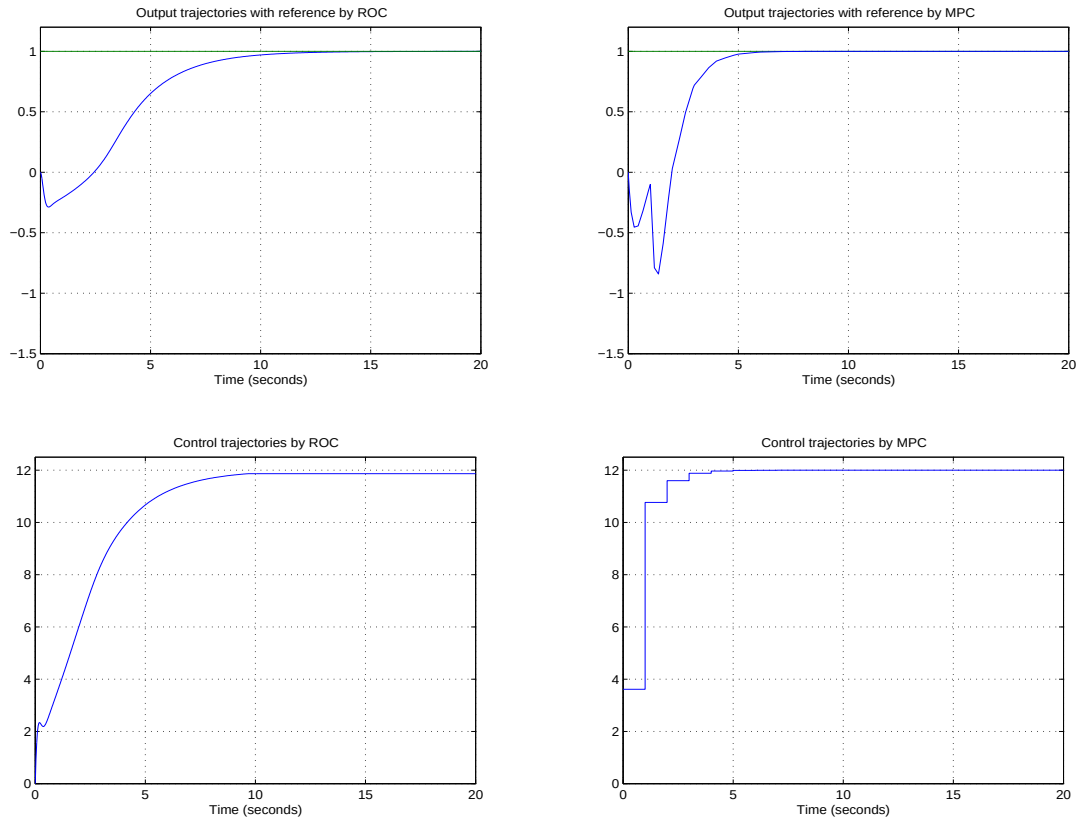


Figure 3.5: Non-minimum phase system with constraints controlled by ROC and MPC

It can be seen clearly that the system is getting slower and the peak is tried to be avoided in ROC control. The result shows that the inverse peak is reduced obviously, however

it cannot be totally eliminated because then the controller should cancel the right hand side poles of the system and this would make the closed loop system internally unstable. In the commercial MPC-package, the hard output constraint is violated quite obviously although the system tries to decrease the inverse peak, and as a matter of fact, the inverse peak cannot be avoided easily in general MPC control. It can be also noticed in Figure 3.5 that the control signal generated some backward response in order to get bigger steps to avoid the inverse peaks in ROC control, but because the weight of the control derivative is used, the sluggish responses of output and control trajectories are still presented in ROC control.

As the state variable x is not measured in practice. Using Kalman filtering techniques on the extended disturbance model of the non-minimum phase system from input-output measurements, to estimate the state variables $\hat{x}(k|k)$. The covariance tuning parameters used are $Q_{kal} = 0.01$ and $R_{kal} = 0.01$. The initial state of the plant is $x_0 = [0,\ 0]$, and the estimator is $\hat{x}(k|k) = [0,\ 0]$. The input disturbance is zero-mean white Gaussian distributed noise with variance equals to 0.05, and output disturbance is zero-mean white Gaussian distributed noise with variance equals to 0.1. Then the optimal input trajectory and output trajectory are plotted as Figure 3.6.

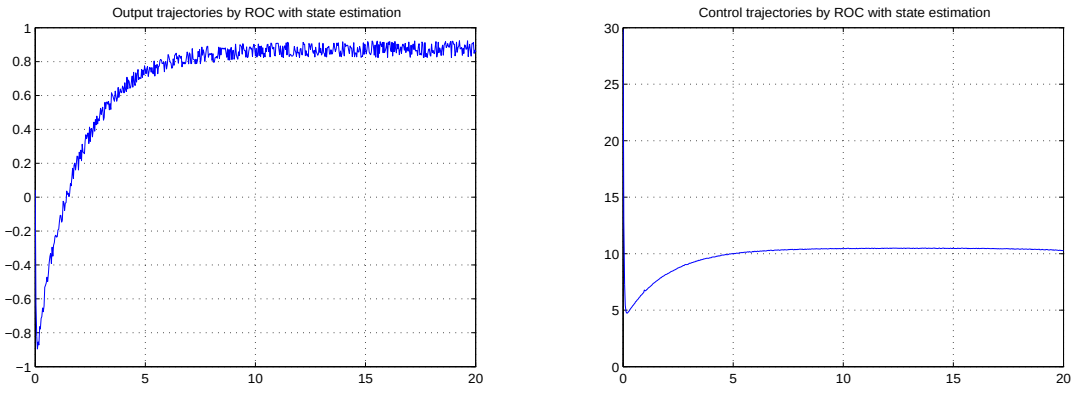


Figure 3.6: Non-minimum phase system controlled by ROC with state estimation

The pair (C, A) describing the overall state-space realization of the combination of plant

and disturbance models has to be detectable. The current state estimates are obtained from the Kalman filter, where the model has been converted to discrete-time state-space form, and the ROC algorithm uses these estimates in future state and controlled output predictions.

Concluding Remarks: The numerical examples in this section are implemented in both the ROC algorithm and the general MPC. The comparison between them is discussed from results. They show clearly that although both ROC and general MPC are the on-line open-loop optimal prediction control approaches, they are quite different control algorithms from the way of derivation, from the predictive model to be handled, from the cost function to be formulated and from the optimization freedoms to be concerned.

Chapter 4

Back-and-Forth Shooting Method

In this chapter, we will present an efficient method called back-and-forth shooting. The solution of TPBVP-form optimization problems derived from the optimal control problems is based on this method. Back-and-forth shooting, originally proposed and named by [OL76], is based on the invariant embedding principle and certain Riccati-type transformations which reduce the TPBVP approximately to the consecutive initial-value problems of ODEs. In these kinds of problems the TPBVP has a special form. Because the initial state should be known, half of the differential equations about state have fixed boundary conditions at the starting point. If the state is not fixed at the end, which is typical in most optimal control problems, the other half of the differential equations about adjoint state (or costate) is fixed at the endpoint, this feature is utilized in the method. Since then the method has been applied with a success to various problems of dynamical optimization, see [Eir85] and resulted in being one of the most efficient alternatives in solving the difficult boundary-value problems.

The chapter starts with a brief introduction to the optimal control problem in a TPBVP form and a scheme for generating the back-and-forth shooting method in an iterative principle is described; after that the detailed algorithm of this method has been specified and its convergence property has been presented; at the end, some numerical examples are given as good illustrations.

4.1 Introduction

As we all know, conventionally an optimal control problem can commonly be presented as an optimization problem with a TPBVP form via Pontryagin's minimum principle [PBGM62]. Consider a linear or nonlinear TPBVP which has a general form:

$$\dot{x}(t) = f(t, x(t), p(t)) \tag{4.1}$$

$$\dot{p}(t) = g(t, x(t), p(t)) \tag{4.2}$$

for $t \in [t_0, t_f]$, subject to the boundary conditions

$$h(x(t_0), p(t_0)) = 0 \tag{4.3}$$

$$e(x(t_f), p(t_f)) = 0 \tag{4.4}$$

where f and h are continuous n-vector-value mappings, g and e are continuous m-vector-value mappings, t_0 and t_f are the fixed initial and final times. Many problems, e.g., in the optimal control and estimation theory, can be reduced to this form via variational calculus or Portryagin's maximum principle.

The numerical solution to TPBVP of ODEs has been studied widely during the period of highly increasing computation facilities, under the needs of optimal control problems, including simple shooting, multiple shooting, collocation and finite difference methods. The simple shooting is to convert the optimization problem into an initial-value problem of ODEs. The ODEs are integrated from t_0 to t_f, and the initial values are varied in order to satisfy the desired end conditions [RVD03a]. The advantage is that the derived solution is a continuously differentiable function, but it has a difficulty to converge for problems whose solutions are very sensitive to initial conditions [RVD03b];

Then a more reliable method, multiple shooting, is developed. This method does not integrate the equations over the full interval in one step, instead it is applied to split up the whole time interval $[t_0,\ t_f]$ into several smaller segments [AMR95]. The "segment"

from an adaptive shooting node on the reference path is solved continuously for numerical integration, and the integration is stopped when the segment exceeds a tolerance value. Then, one starts the integration again from the next adaptive shooting node on the reference path and the previous step is repeated, until the system is integrated to the final time t_f [RVD03a]. An advantage of this method is that the convergence can be obtained for a larger class of TPBVPs [AMR95], however its disadvantage is that the number of parameters to be updated in each iteration can be very large, causing fairly large computation times [RVD03a], and during each iteration, one has to invert matrices row and column dimensions of which are linear functions of the number of shooting nodes. The number of nodes can be quite large depending on the guesses for the initial estimates [RVD03b]. Another disadvantage of this method is that if the differential equations are re-integrated to result in one continuous trajectory for the system, the actual final values may not be close to the desired final values. This is a common problem when solving TPBVPs that result from optimal control, due to instability of the systems in the forward direction [AMR95].

The collocation and finite difference methods are far more complex to set up. Some good discussion about these two approaches can be also found in [AMR95]. However *bvp4c* algorithm in MATLAB R14 based on the routine of collocation method will be implemented into some numerical examples as a comparison method to the back-and-forth shooting.

In 1976, [OL76] tried to develop an iterative method from the Equations (4.1)-(4.4) via a minimal number of approximations, which adopts the merits from the multiple shooting method but avoids the instability of the systems in the forward direction by shifting the integration in both the backward and forward direction. They succeeded very well and ended up with an method called ***back-and-forth shooting***, a very effective method we will rely upon in this thesis work for solving TPBVPs derived from general optimal control problems.

Algorithm Scheme 4.1 *The optimal problem Equations (4.1)-(4.4) is considered as the following primitive iteration scheme at $t \in [t_0, t_f]$ in this thesis fitting for the time-varying system [OL76]:*

$$\dot{x}_{i+1}(t) = f(t, x_{i+1}(t), p_i(t)), \qquad h(x_{i+1}(t_0), p_i(t_0)) = 0 \tag{4.5}$$

$$\dot{p}_{i+1}(t) = g(t, x_{i+1}(t), p_{i+1}(t)), \quad e(x_{i+1}(t_f), p_{i+1}(t_f)) = 0 \tag{4.6}$$

for the iteration indices $i \geq 0$, based on a suitable initial guess p_0 (or alternatively, Equations (4.5) and (4.6) in reverse order, based on an initial x_1).

4.2 Back-and-Forth Shooting Algorithm

Roughly speaking, this method involves the forward integration of Equation (4.1) and the backward integration of Equation (4.2), by turns, with the aid of certain correction terms and auxiliary functions. Suppose that pair $(\bar{x}, \bar{p})$ is a desired solution of the problem Equations (4.1)-(4.4) and that an initial guess for it is available, then based on the primitive iteration **Algorithm Scheme 4.1**, the algorithm proposed by [OL76] can be given as follows:

Algorithm Scheme 4.2 (back-and-forth shooting)

[*1*] Given an initial guess (x_0, p_0), set the initial index $i = 0$.

[*2*] By numerical integration solve the following initial-value problem of the Riccati equation, for the $n \times m$ matrix valued function K_i on $t \in [t_0, t_f]$:

$$\begin{aligned} \dot{K}_i(t) = f_x(t, x_i(t), p_i(t))K_i(t) - K_i(t)g_p(t, x_i(t), p_i(t)) \\ - K_i(t)g_x(t, x_i(t), p_i(t)) + f_p(t, x_i(t), p_i(t)), \end{aligned} \tag{4.7}$$

with

$$K_i(t_0) = -h_x(x_i(t_0), p_i(t_0))^{-1} h_p(x_i(t_0), p_i(t_0)), \tag{4.8}$$

then store K_i.

[*3*] Solve the following n-vector initial-value problem for r_{i+1} over $[t_0, t_f]$:

$$\dot{r}_{i+1}(t) = f(t, r_{i+1}(t), p_i(t)) - K_i(t)\left[g(t, r_{i+1}(t), p_i(t)) - \dot{p}_i(t)\right], \tag{4.9}$$

with

$$h(r_{i+1}(t_0), p_i(t_0)) = 0, \tag{4.10}$$

then store r_{i+1}.

[*4*] By numerical integration backwards solve m-vector terminal-value problem for p_{i+1} over $[t_0, t_f]$:

$$\dot{p}_{i+1}(t) = g(t, r_{i+1}(t) + K_i(t)\left[p_{i+1}(t) - p_i(t)\right], p_{i+1}(t)), \tag{4.11}$$

with

$$e(r_{i+1}(t_f) + K_i(t_f)\left[p_{i+1}(t_f) - p_i(t_f)\right], p_{i+1}(t_f)) = 0, \tag{4.12}$$

then set

$$x_{i+1}(t) = r_{i+1}(t) + K_i(t)\left[p_{i+1}(t) - p_i(t)\right], \tag{4.13}$$

then store x_{i+1} and p_{i+1}.

[*5*] If the difference between p_{i+1} and p_i is less than a prescribed tolerance, Stop. Otherwise replace i by $i+1$, and repeat steps [*2*] -[*5*].

To let this algorithm work, some solvability conditions must be fulfilled: $r_{i+1}(t_0)$ can be solved uniquely from Equation (4.10) and also $p_{i+1}(t_f)$ can be solved uniquely from Equation (4.12) [OL76]. Note that with regard to Equation (4.12), the present **Algorithm Scheme 4.2** is also applicable to fixed-endpoint problems in optimal control, if the matrix $K_i(t)$ is invertible [OL76]. Most of the nonlinearity properties of Equations (4.1)-(4.4) are maintained in the algorithm, the only approximation is for the determination of K_i's by the Riccati equation in the linearized case [OL76].

Generally speaking, the main steps when applying the back-and-forth shooting method in the numerical solution of optimal control problems are: the derivation of the necessary conditions (e.g., the adjoint state differential equations); the estimation of an appropriate initial guess of the unknown state and adjoint variables in order to start the iteration process, and a good initial estimation of optimal solution is very necessary for fast convergence, especially in some nonlinear constrained optimal control problems, e.g. cascaded hydro-electric power plant chain system, which will be studied deeply in the next chapter. The great advantage of the back-and-forth shooting method is that the adjoint state differential equations are re-integrated to result a continuous trajectory for the system in the backward direction when instability of the systems in the forward direction is caused, then a stability of the systems in the backward direction can be guaranteed.

4.3 Algorithm Convergence

The complete convergence analysis or detail proofs of the method will not be performed here, only some main results are reviewed. For further information, we refer the readers to [Eir83, Eir85]. It is easy to see that if the sequence (x_i, p_i) generated by the back-and-forth shooting method convergences uniformly and the K_i sequence is bounded, then the limit will be a solution for Equations (4.1)-(4.4). It is approved in [Eir85]:

Theorem 4.1 *Let f, g, h, e be twice continuously differentiable. Suppose that $(\bar{x}, \bar{p})$ is a solution to optimal problem Equations (4.1)-(4.4), such that*

- *$h_x(\bar{x}(0), \bar{p}(0))$ is invertible;*
- *Riccati Equation (4.7), with initial condition Equation (4.8), has a solution $\bar{K}$ along $(\bar{x}, \bar{p})$ on $[t_0, t_f]$;*
- *$e_x(\bar{x}(t_f), \bar{p}(t_f))\bar{K}(t_f) + e_p(\bar{x}(t_f), \bar{p}(t_f))$ is invertible.*

Then the sequence $\langle (x_i, p_i) \rangle_{i=0}^{\infty}$ generated by the back-and-forth shooting method converges to $(\bar{x}, \bar{p})$, provided (x_0, p_0) is sufficiently close to $(\bar{x}, \bar{p})$.

Under the assumption of Theorem 4.1, the norm of the convergence is firstly defined by [Eir85] as:

$$\|(x,p)\| = |x(0)| + |p(0)| + sup_{t\in[t_0,t_f]}(|\dot{x}(t)| + |\dot{p}(t)|) \tag{4.14}$$

where $|\cdot|$ is the Euclidean norm. Further, if the second derivatives of f, g, h and e are uniformly Lipschitzian in x and p argument positions, in a neighborhood of the trajectory of $(\bar{x},\bar{p})$, then there exists M such that, for all $i > 0$, the convergence is quadratic [Eir85], i.e.

$$\|(x_i,p_i) - (\bar{x},\bar{p})\| \leq M \|(x_{i-1},p_{i-1}) - (\bar{x},\bar{p})\|^2 \tag{4.15}$$

If the assumptions of the previous theorem is fulfilled, then for linear problems (f, g, h and e are linear) one-step convergence is guaranteed, i.e., $(x_i,p_i) = (\bar{x},\bar{p})$ for any (x_{t_0},p_{t_0}). The proof of this local convergence theorem is based on the Fréchet differentiability of the mapping assigning (x_{i+1},p_{i+1}) to (x_i,p_i) and a version of the fixed point theorem. Furthermore, the rate of convergence is also shown to be quadratic, which is the best way achieved with linear approximation [Eir85]. Once this numerical method has converged, the norm of the difference between the desired final adjoint states and the actual final adjoint states obtained after a re-integration of the equation is our criterion for accuracy of the solution.

4.4 Numerical Examples

Example 4.1: The example to be considered here is a continuous-time MIMO model from a linear dynamic system, its transfer function is given as:

$$g(s) = \begin{bmatrix} \frac{0.5s+1}{2s^2+3s+1} & \frac{2s+1}{s^2+2s+1} \\ \frac{0.4}{3s+1} & \frac{1}{2s+1} \end{bmatrix} \tag{4.16}$$

with the cost function forms:

$$J = \int_0^{t_f} \Big(x^T Q x + u^T R u\Big) dt + x^T(t_f) F x(t_f) \tag{4.17}$$

where $t_f = 25$ is final time. R is the weight matrix of inputs:

$$R = \begin{bmatrix} 0.1000 & 0 \\ 0 & 0.1000 \end{bmatrix} \tag{4.18}$$

and Q is weight matrix of system outputs:

$$Q = \begin{bmatrix} 0.6250 & 1.2500 & 0 & 2.5000 & 1.2500 & 0 \\ 1.2500 & 2.5000 & 0 & 5.0000 & 2.5000 & 0 \\ 0 & 0 & 0.7111 & 0 & 0 & 2.6667 \\ 2.5000 & 5.0000 & 0 & 10.0000 & 5.0000 & 0 \\ & 1.2500 & 2.5000 & 0 & 5.0000 & 2.5000 \quad 0 \\ 0 & 0 & 2.6667 & 0 & 0 & 10.0000 \end{bmatrix} \tag{4.19}$$

and F is the weight matrix of terminal states:

$$F = \begin{bmatrix} 0.1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1000 \end{bmatrix} \tag{4.20}$$

Then a reduced linear TPBVP via Pontryagin's principle is presented below:

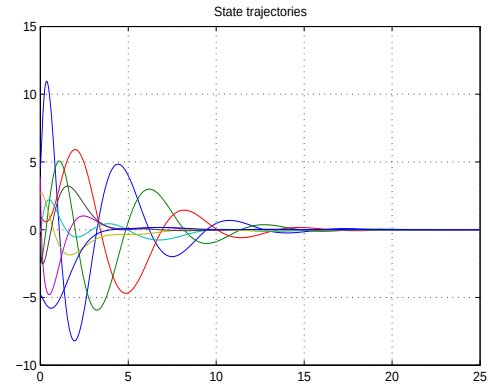
$$\dot{x} = Ax - B(R + R^T)^{-1}B^T p \tag{4.21}$$

$$\dot{p} = -(Q + Q^T)x - A^T p \tag{4.22}$$

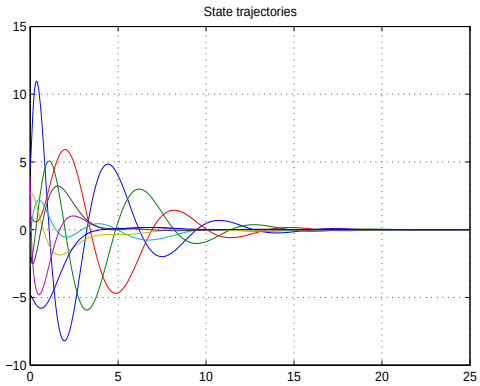
for $t \in [0, t_f]$, subject to the boundary conditions

$$x(0) = x_0, \quad p(t_f) = (F + F^T)x(t_f), \tag{4.23}$$

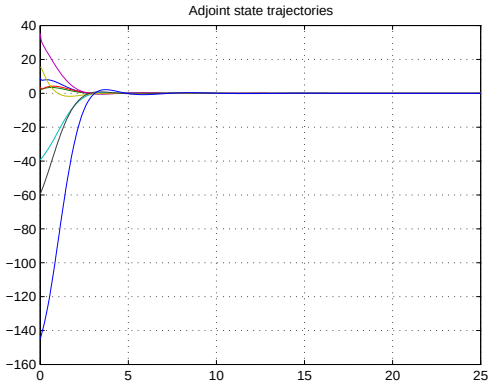
via the back-and-forth shooting ***Algorithm 4.2***, above TPBVP problem is solved with the results illustrated in Figure 4.1. Since the system has linear dynamics, only one iteration is needed for obtaining the solution. The accuracy $\|p_{i+1} - p_i\|$ is better than six decimal digits.



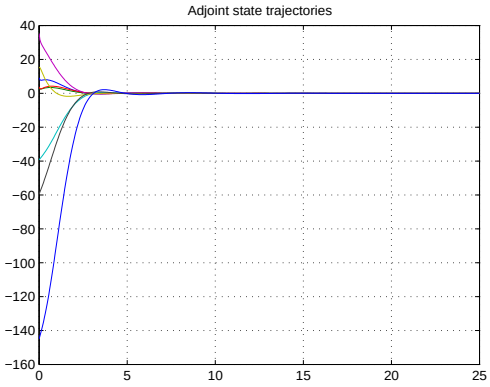
(a) back-and-forth shooting



(b) *bvp4c*



(c) back-and-forth shooting



(d) *bvp4c*

Figure 4.1: Solution of a quadratic optimal control problem

As a comparison, the typical algorithm for solving boundary-value problem method *bvp4c* in MATLAB is also implied. Under same accuracy requirement, *bvp4c* algorithm obtains the same result as the back-and-forth shooting algorithm does.

Example 4.2: Consider a simplified model from [OL76], a daily optimal discharge control of a hydroelectric power plant (time constants are expressed in days in this case):

$$\dot{x}(t) = 3 - u(t), \quad x(0) = 1, \tag{4.24}$$

The variable x is the volume of the water storage in the forebay, and the variable u is the discharge through the turbines. The possible is to determine a control variable u on $[0, T]$ and the corresponding motion x which minimize the cost function

$$J = \int_0^T \Big\{ -b(t) \left[1 + 0.2x(t) - 0.025u(t) \right] u(t) + 50 \left[x(t) - 0.5 \right]^{10} \Big\} dt - x(T) \tag{4.25}$$

where

$$b(t) = 1 - a \cos(6.28318t), \quad T = 3.25, \tag{4.26}$$

where the variable b is the price of the electric power generated, and a is a constant nonnegative parameter with value $0, \ldots, 0.7$. The period length 1 corresponds to one day [OL76]. The expression within the first brackets in Equation (4.25) represents the net pressure of water (or head) in the turbines; e.g. the term of $-0.025u(t)$ represents roughly the head losses due to the tailwater flow. The latter term in the integrand is an artificial penalty cost, where the approximate constraints $0 \leq x \leq 1$ are taken into account as $|x(t) - 0.5| \leq 0.5$. Then the optimization problem in TPBVP is obtained at $t \in [0, T]$:

$$\dot{x}(t) = -4x(t) - [20/b(t)]p(t) - 17 \tag{4.27}$$

$$\dot{p}(t) = 0.8b(t)x(t) + 4p(t) + 4b(t) - 500[x(t) - 0.5]^9 \tag{4.28}$$

subject to boundary conditions

$$x(0) = 1, \quad p(T) = -1, \tag{4.29}$$

Equation (4.28) is highly nonlinear with regard to its last term. Following the back-and-forth shooting ***Algorithm 4.2***, algorithm above has been used and K_i's in the algorithm

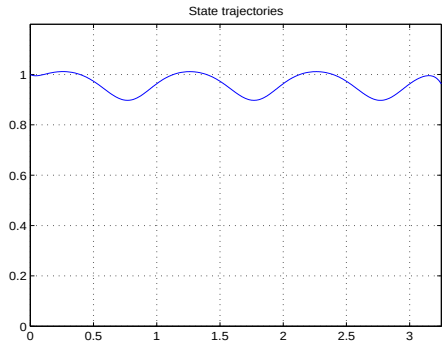
are given by the differential equation:

$$\dot{K}_i(t) = -8K_i(t) - \left(0.8b(t) - 4500(x_i(t) - 0.5)^8\right) K_i^2(t) - \frac{20}{b(t)} \tag{4.30}$$

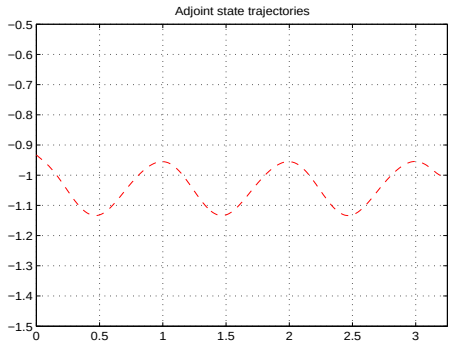
where $K_i(t) = 0$, the differential equations are integrated using fixed-step fourth-order Runge-Kutta method (ODE4) [ODE] with the fixed time stepsize 0.00625. Results are illustrated in Figure 4.2. Varied with different values of a, the time function of state trajectories x are shown on the left side, and the adjoint state trajectories p are on the right side of the figure.

The behavior of the solutions is satisfied and the constraints requirements of state constraints are fulfilled quite well, and all results were the same as in [OL76]. Figure 4.2 gives the solution of Equations (4.27) - (4.29) only after five iterations, so the convergence is quite fast and the accuracy $\|p_{i+1} - p_i\|$ is better than six decimal digits. As a comparison, the *bvp4c* algorithm is also implemented, however, the solution unfortunately overflows because of the high nonlinearity from the last term of Equation(4.28).

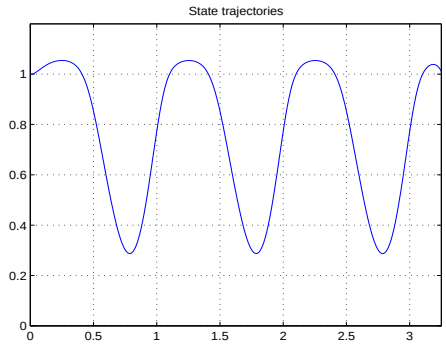
Concluding Remarks: The entire computations were performed on the CPU 2.8GHz and the algorithm was programmed in MATLAB. In these two preliminary test examples, the algorithm gives very promising results. The accuracy $\|p_{i+1} - p_i\|$ is better than six decimal digits. The convergence speed is very fast: one iteration for the first linear example with the computation time is about 0.007 seconds; and only five iterations are needed for second nonlinear example with computation time is about 0.056 seconds to achieve the solution of the problems. As a matter of fact, the experiments also showed that in a case like the second example, the conventional shooting methods cannot solve it reasonably. Therefore in the next chapter, this proposed back-and-forth-shooting method will be used in the ROC algorithm for handling all TPBVP-form optimal control problems, some of them are quite complicated so that the general MPC with its related optimal control algorithms may be unable to solve easily in both cases.



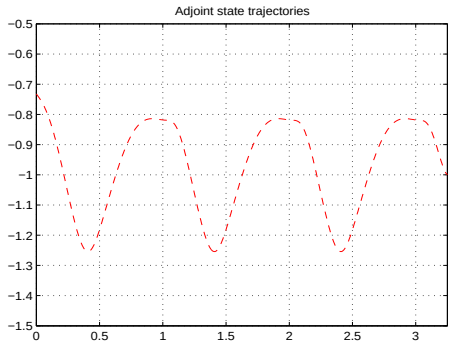
(a) parameter a = 0.1



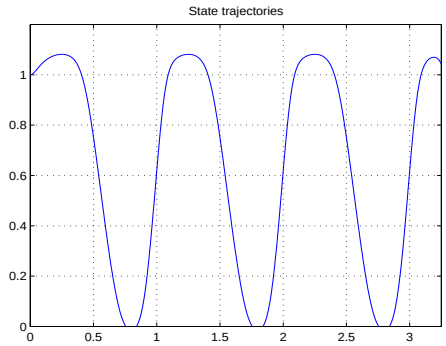
(b) parameter a = 0.1



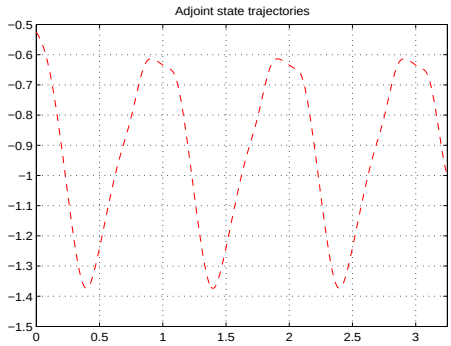
(c) parameter a = 0.3



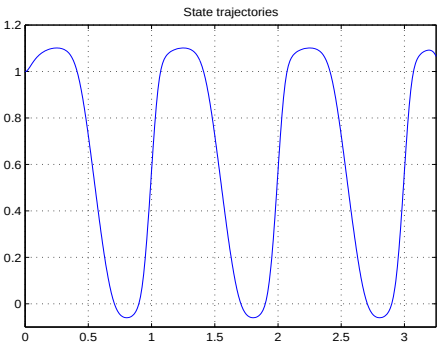
(d) parameter a = 0.3



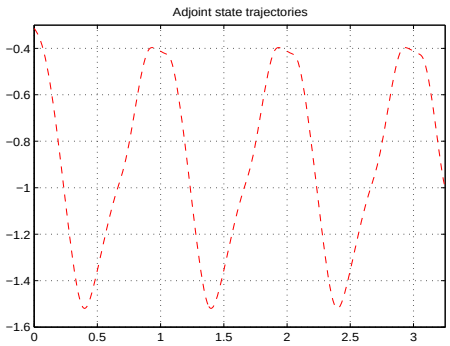
(e) parameter a = 0.5



(f) parameter a = 0.5



(g) parameter a = 0.7



(h) parameter a = 0.7

Figure 4.2: Solution of a simplified hydro-power plant model

Chapter 5

Semi-analytical Solution to Applications via ROC control

In this chapter, the ROC algorithm is implemented to two application examples. The first one is a cascaded hydro-electric power plant chain, which has the most difficult form for solving the optimal control problem as we mentioned at the beginning of Chapter 3, the fixed end point issue. The application system dealt with is dynamically nonlinear, and the application study involves multivariable control, state and input constraint handling, and optimization. The application study is mainly used to testify the powerful capability of ROC algorithm for dealing fairly difficult and complex optimal control problem and at the same time explicitly handling multiple constrained states and inputs very well. The second application is a multivariable nonlinear reactor with two inputs, two outputs and multiple steady states. It is intended to show how the ROC algorithm works in some industrial-like control problems. All application algorithms are implemented in the MATLAB environment.

The chapter starts with a brief introduction to the first application example and a dynamic mathematical model is built for testing or demo purpose. The controlled hydropower plant chain problem is then formulated as an optimal control problem reduced into TPBVP-form. The back-and-forth shooting method to TPBVP forms an "inner loop" for differential solution and ROC optimization control algorithm forms an "outer loop"

for optimal control of the system. The results are presented and the algorithm control capability is proved. Then a more industrial practical example is given, the application is a two-dimensional exothermic chemical reactor.

5.1 Cascaded Hydro-electric Power Plant Chain

5.1.1 Introduction

The application example used here is referred to the part of cascaded hydro-electric power plant chain along the river Oulujoki in Finland given in [Lau79]. It is mainly taken for testing or demo purpose. The map and corresponding height profile of the cascaded hydro-electric power plant chain is depicted in Figure 5.1:

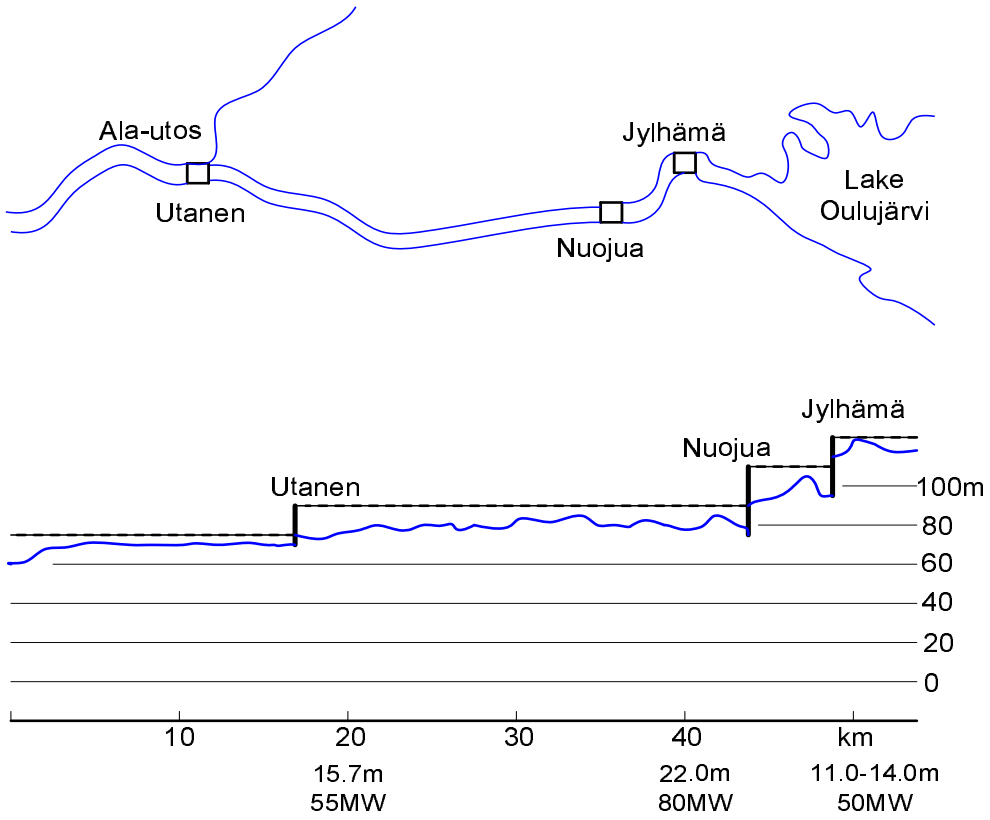


Figure 5.1: Map and height profile of a part of the river Oulujoki

The average discharge of one plant is about 230 m^3/s, and every plant has three turbines with maximum total discharge of 450 m^3/s. The heads of the plants are relatively low as

can be seen in the figure, and the maximum power of the system is 185 MW. The areas of the forebays in the river are only some quadrat-kilometers and the allowable ranges of the level heights are 2.2 m and 1.25 m. This means that the ranges may be crossed during few hours even in quite reasonable operation of the chain. It also means that in daily operation the level constraints must be taken into account carefully, and this difficulty is also present in the optimization. On the contrary, the first forebay in the cascaded chains, Lake Oulujärvi is so big that its level can be considered as fixed [Lau79].

5.1.2 Mathematical Model

When building the model, the following assumptions had been done in [Lau79]: the rate of change of the discharge is so low, that the acceleration forces of flow can be neglected, the friction forces follow the Manning formula, and the reservoire have vertical walls in the level variations in question. In addition, by zero discharge the level in the forebay can be at the same height as the tailwater of the preceding hydro-plant. The cascaded hydro-electric power plant chain system in Figure 5.1, is then approximated to a lumped parameter model by [Lau79] illustrated in Figure 5.2:

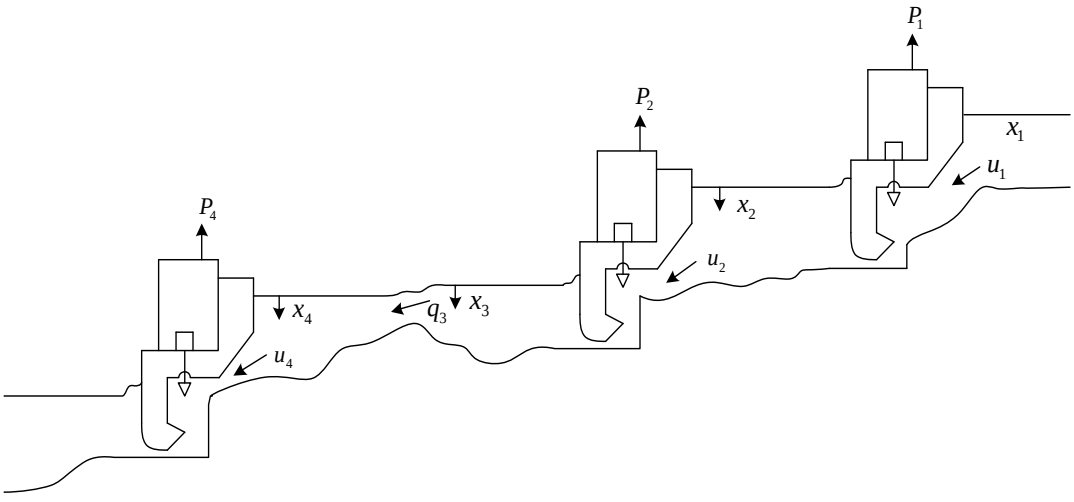


Figure 5.2: Model of the hydro-electric power plant chain

The P_i are active powers, u_i are discharges, q_3 is a flow and x_i are the height deficits of the water levels except x_1, which is a volume variable. The model equations can now be derived from mass balance, friction law and experimentally measured functions. The

storage equations are given by [Lau79]

$$\begin{aligned}
\dot{x}_1 &= u_1 \\
\dot{x}_2 &= a_2(u_2 - u_1) \\
\dot{x}_3 &= a_3\big(\sqrt{g(x_4 - x_3)[1 - h(x_3 + x_4)]} - u_2\big) \\
\dot{x}_4 &= a_4\big(u_4 - \sqrt{g(x_4 - x_3)[1 - h(x_3 + x_4)]}\big)
\end{aligned} \tag{5.1}$$

where a_i is the inverse of the effective area of the storage. The experimentally defined power conversion function includes also the head loss in the tailwater channel, and then the overall active power conversion function of each plant is given by [Lau79]

$$\begin{aligned}
P = &-e_1 - e_2 - e_4 + c_1u_1 + c_2u_2 + c_4u_4 \\
&- d_1u_1^2 - d_2u_2^2 - d_4u_4^2 + k_1x_2u_1 + k_2(x_3 - x_2)u_2 - k_4x_4u_4
\end{aligned} \tag{5.2}$$

For the sake of convenience, the state variable of Equations (5.1) and (5.2) are scaled in [Lau79] as: The unit of the time is 1 hour, and the unit of the flow rates is the maximum discharge 450 m^3/s. By preserving the first equation in (5.1), the unit of the volume variable x_1 corresponds to 1,620,000 m^3, i.e., the water amount by the maximum discharge during an hour. The unit of the deficit variables x_2, x_3, and x_4 is 1 meter. The unit of the power P is 1 MW and thereby the following values of parameters are used:

$$a_2 = 1.1, \quad a_3 = 0.55, \quad a_4 = 0.55, \quad g = 0.67, \quad h = 0.4, \tag{5.3}$$

$$e_1 = 0.7, \quad e_2 = 1.3, \quad e_4 = 1.4, \quad c_1 = 55, \quad c_2 = 96, \quad c_4 = 70, \tag{5.4}$$

$$d_1 = 10, \quad d_2 = 8.3, \quad d_4 = 13.3, \quad k_1 = 4.05, \quad k_2 = 3.74, \quad k_4 = 3.40. \tag{5.5}$$

The "river parameters" like a_i, g and h are chosen after physical realities as areas of the water surfaces and measured friction heights in some typical operation conditions.

5.1.3 Optimization Problem Formulation

The whole power generating system has been arranged in a way that the water consumption in the optimization is fixed, and a marginal price curve is taken into consideration

and the optimization problem is then formulated as follows [LO76, Lau79] :

$$
\begin{aligned}
\dot{x}_1 &= u_1 \\
\dot{x}_2 &= a_2(u_2 - u_1) \\
\dot{x}_3 &= a_3\big(\sqrt{g(x_4 - x_3)[1 - h(x_3 + x_4)]} - u_2\big) \\
\dot{x}_4 &= a_4\big(u_4 - \sqrt{g(x_4 - x_3)[1 - h(x_3 + x_4)]}\big) \\
x(t_0) &= y_1, \quad x(t_f) = y_2, \\
0 &\le x_2(t) \le x_{2max} = 2.2, \\
0 &\le x_4(t) \le x_{4max} = 1.25, \\
0 &< u_{imin} \le u_i(t) \le u_{imax} \le 1, \quad i = 1, 2, 4, \\
J_1 &= \textstyle\int_{t_0}^{t_f} \big[-b(t)P(x(t), u(t))\big]\, dt, \\
P &= -e_1 - e_2 - e_4 + c_1u_1 + c_2u_2 + c_4u_4 \\
&\quad -d_1u_1^2 - d_2u_2^2 - d_4u_4^2 + k_1x_2u_1 + k_2(x_3 - x_2)u_2 - k_4x_4u_4
\end{aligned}
\tag{5.6}
$$

where x denotes the state vector, y_1 and y_2 are fixed initial and terminal states, $b(t)$ is a given price curve and J_1 is the cost function to be minimized. The range constraints on the water storages, which are here state quantities, poses a difficult question on the implementation of solution methods for the equation induced by variational calculus [LO76]. Here the state constraints are relaxed by a method of penalty functions. It is a well-known approach, where the original cost function is replaced by an augmented cost function [Lau79]. In this case

$$
J = J_1 + \int_{t_0}^{t_f} \left\{ s_2 \left[2\frac{x_2(t)}{x_{2max}} - 1 \right]^N + s_4 \left[2\frac{x_4(t)}{x_{4max}} - 1 \right]^N \right\} dt \tag{5.7}
$$

where s_i are "small" positive constants and N is a penalty exponent with even integer value. In general, there are many possibilities to choose the form of the penalty terms. The augmented part in expression (5.7) gives very small contribution to the integral if the state constraints are not violated, and the contribution increased rapidly, if violations occur. This feature is common for all penalty functions. It is then important that in the optimum obtained the contribution of the penalty terms is negligible. In this case a high value of N and low values of s_i's guarantee a good approximation of the solution of the

original problem [LO76]. Anyhow, when a candidate for the optimum is available, the correctness of the solution can easily be checked. With the given boundary conditions, the dynamical equations and the state and control constraints, etc, the above optimization problem can now be reduced to a TPBVP via the Pontryagin's minimum principle. In a detailed expression, the TPBVP is given as:

$$\begin{aligned}
\dot{x}_1 &= u_1^* \\
\dot{x}_2 &= a_2(u_2^* - u_1^*) \\
\dot{x}_3 &= a_3\big(\sqrt{g(x_4 - x_3)[1 - h(x_3 + x_4)]} - u_2^*\big) \\
\dot{x}_4 &= a_4\big(u_4^* - \sqrt{g(x_4 - x_3)[1 - h(x_3 + x_4)]}\big) \\
\dot{p}_1 &= 0 \\
\dot{p}_2 &= bk_1u_1^* - bk_2u_2^* - \tfrac{2Ns_2}{x_{2max}}\left(\tfrac{x_2(t)}{x_{2max}} - 1\right)^{N-1} \\
\dot{p}_3 &= bk_2u_2^* + (p_4a_4 - p_3a_3)(2ghx_3 - g)/(2\sqrt{g(x_4 - x_3)[1 - h(x_3 + x_4)]}) \\
\dot{p}_4 &= -bk_4u_4^* + (p_4a_4 - p_3a_3)(g - 2ghx_4)/(2\sqrt{g(x_4 - x_3)[1 - h(x_3 + x_4)]}) \\
&\quad -\tfrac{2Ns_4}{x_{4max}}\left(\tfrac{x_4(t)}{x_{4max}} - 1\right)^{N-1}
\end{aligned} \tag{5.8}$$

where

$$u_i^* = \begin{cases} u_{imin} & \text{if } u_i^o < u_{imin} \ , \\ u_i^o & \text{if } u_{imin} \le u_i^o \le u_{imax} \ , \\ u_{imax} & \text{if } u_i^o > u_{imax} \ , \ i = 1, 2, 4; \end{cases} \tag{5.9}$$

$$\begin{aligned}
u_1^o &= \big(c_1 + k_1x_2 - p_1/b + a_2p_2/b\big)/2d_1 \\
u_2^o &= \big(c_2 + k_2(x_3 - x_2) - a_2p_2/b + a_3p_3/b\big)/2d_2 \\
u_4^o &= \big(c_4 + k_4x_4 - a_4p_4/b\big)/2d_4
\end{aligned} \tag{5.10}$$

subject to the one of the most difficult boundary condition forms in optimal control problem: the fixed terminal state $x(t_f)$, so

$$x(t_0) = y_1, \quad x(t_f) = y_2, \tag{5.11}$$

However most optimal control problems in real industry can be described using free end points formulation, in that conditions, the terminal state $x(t_f)$ is not fixed. The following special form of TPBVP mentioned in previous chapters, the adjoint state $p(t_f)$ has to be fixed at the endpoint instead, and the boundary conditions turn to be

$$x(t_0) = y_1, \quad p(t_f) = y_3 \tag{5.12}$$

In the later section of this chapter, numerical experiments based on both of these two different forms of TPBVP will be studied for ROC algorithm testing purpose.

5.1.4 Semi-analytical Solutions via ROC Control

In this section, we implement ROC control into this hydro-power plant system, by solving its TPBVP form on-line open-loop optimal control problem Equations (5.8)-(5.11) repetitively with the aid of state (and disturbance) estimation and obtain the required constrained optimal control trajectory.

We consider a sequence of sampling instants $\{t_i\}_{i\geq 0}$ with a constant sampling interval $t_s > 0$ (note that t_s is set much smaller than the prediction horizon H_p, i.e. $t_s << H_p$), so that $t_{i+1} = t_i + t_s$ for all $i \geq 0$. Then ROC control solve repeatedly the on-line open-loop optimal control problem Equations (5.8)-(5.11) at each sampling instant t_i. In every iteration using the current state estimations of the plant $x(t_i)$ to compute the optimal controls $u^*(t_i, x(t_i))$ from equations (5.9) and (5.10). The measurable control function u and the corresponding state trajectory x, which satisfy the constraints of optimal control problem, are obtained from the model of the system. x^o and u^o denote the state trajectory and optimal control solution to an open-loop optimal control problem, and x^* and u^* are the optimal state trajectory and control resulted from the ROC control scheme.

Illustrated in Figure 5.3, the following steps in controlling of the hydro-electric power plant chain system with ROC strategy are applied:

[*1*] estimate the first initial states $x(t_i)$ of the plant at the initial time t_i (current ini-

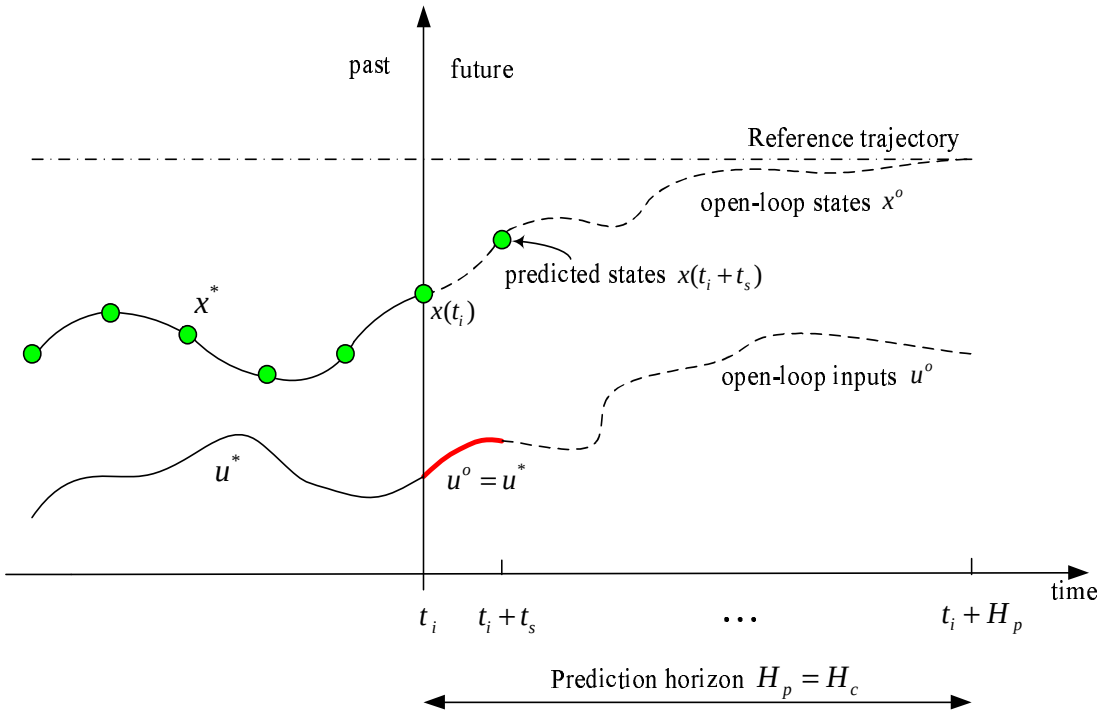


Figure 5.3: Principle of ROC control for hydro-electric power plant chain

tial index $i = 0$). Note that although in practice, the measurements of states are not available, this hydro-electric power plant chain system is an application mainly used for ROC algorithm testing purpose, so we suppose that the initial states can be measured in the rest of repetition, but in practice, the measurements of states are not always available.

Furthermore, a good guess of the first initial states is very important for fast convergence, especially in this nonlinear hydro-power plant chain system, which is very difficult to be solved by normal optimization method, even the back-and-forth shooting method needs a quite moderate first initial state guess to solve it, and the rest of initial guesses are always suitable for optimization solution according to the **Theorem 3.1** Belman's optimal principle [Bel57];

[*2*] compute the open-loop optimal controls u^o from the reduced TPVBP-form optimization problem at $[t_i, t_i + H_p) \rightarrow R^n$ (current initial index $i = 0$), with the constraint requirements fulfilled;

[*3*] apply the optimal controls $u^* = u^o$ at the time interval $[t_i, t_i + t_s)$ to the hydro-electric power plant chain system;

[*4*] repeat the procedure from step [1] for the next sampling instant t_{i+1} (the index i is incremented by one unit in each repetition), and prediction horizon moves forwardly one t_s time-step further.

Therefore a semi-analytical solution of feedback control is obtained during the each sampling interval, as a function of states at each sampling instant. The optimal control $u^*(t)$, $t \in [t_i, t_i + t_s)$ is applied until the next sampling instant t_{i+1} is achieved and at that time a new optimal control problem is solved. The optimal control variables are stored on discrete-time instants first, and then approximated to the intermediate time values via *spline interpolation*. The whole procedure which includes both optimization and prediction is repeated to find a new input with the prediction horizons forward-moving, where the back-and-forth shooting solution of TPBVP forms an "inner loop" for optimization and the ROC optimal control algorithm forms an "outer loop" for prediction.

Note that in the back-and-forth shooting optimization problems solution, the TPBVP has a special form: If the state x is not fixed at the end, the other half of the differential equations about adjoint state p has to be fixed at the endpoint and the values of $x(t_0)$ and $p(t_f)$ are used as initial guesses for p and x; If both initial and final values of the states x are fixed, then the initial guess of the final value of $p(t_f)$ is required and the rest final values of adjoint states are calculated in every iteration by

$$p_{i+1}(t) = K_i(t)^{-1} \left[x(t_f)^T - r_{i+1}(t) \right] + p_i(t) \tag{5.13}$$

The whole computations were performed on the CPU 2.8GHz and the algorithm was programmed in MATLAB. The differential equations are integrated by using a modified Runge-Kutta method (ODE4). The price curve $b(t)$ for weekdays used in optimization calculations is depicted by [LO76] as Figure 5.4. The curve repeats the first 24 hours behaviors day by day and only for Saturday it varies only from 90 `FM/MWh` to 125 `FM/MWh` with the same shape and for Sunday it is near constant 90 `FM/MWh` [LO76].

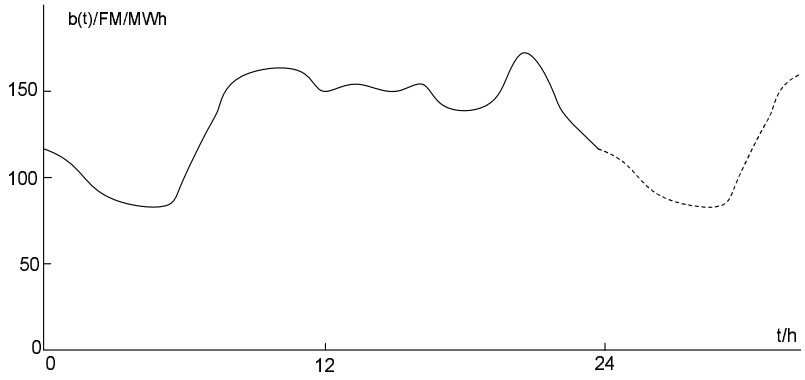


Figure 5.4: Marginal price b for a day

5.1.5 Implementation Cases

In all the following implementation cases, the optimization intervals are chosen to be either one day (24 hours) or one week (168 hours). The sampling time is defined as $T_s = 1$ hour and the prediction horizon is defined the same as its minimum optimization interval length, i.e. $H_p = 24$ hours because in all cases of this application, prediction with nonlinear TPBVP model requires to solve approximately the consecutive initial-value ODE problems. The first initial guess of states (and adjoint states), which are used for starting the algorithm for solving the optimization problem, are defined in each case, respectively. And the rest of the initial guess of states are chosen from the measured states at each iteration. A good initial estimation of the optimal solution is very necessary for fast convergence.

Case 1: In this example, we firstly give a simple optimization problem with a boundary condition that only the initial state is known without a fixed-end point [LO76], so:

$$x(t_0) = [0,\ 1,\ 0.2,\ 0.5]^T, \tag{5.14}$$

$$p(t_f) = [23800,\ 16200,\ 12800,\ 12800]^T \tag{5.15}$$

As we mentioned in the previous chapter, in this kind of optimization problems the TPBVP has a special form. If the state is not fixed at the end, the other half of the differential equations about adjoint state has to be fixed at the endpoint. So in this case

both the initial value of the state and final value of the adjoint state are fixed and these values are also used as initial guesses for p and x.

The estimation of initial states and adjoint states is very important for fast convergence, especially in this nonlinear hydro-power plant chain system, which is very difficult to be solved by normal optimization method, even the back-and-forth shooting method needs the fairly moderate first initial state (and adjoint) guess to solve it, and the rest of initial guesses, which are got from the measurement are always suitable for optimization solution according to the **Theorem 3.1** Belman's optimal principle [Bel57]. For convenience, we adopt the existing good initial guesses of p and x provided in [LO76] for the optimization solution and ROC control in this section. Then the optimal control problem (5.8)-(5.11) is solved according to previous general parameters settings Equations (5.3)-(5.5) and

$$\begin{aligned} u_{imin} &= 0.15, \quad u_{imax} = 0.15, \quad i = 1, 2, 4 \\ T &= 24, \quad N = 30, \quad s_2 = s_3 = 0.5 \end{aligned} \tag{5.16}$$

The result is shown in Figure 5.5 when the norm of the error-array $\|p_{i+1} - p_i\|$ is smaller than 1e-5. The left-side three pictures in Figure 5.5 illustrate an open-loop optimization solution via the back-and-forth shooting method. It demanded 7 iterations and calculation time was about 2.218 seconds for each iteration. Step size was set as 0.25. Results were also compared with my colleague (Juha Lassila)'s and the final results were the same. There is a very slight difference in some elements of the partial derivatives matrices K_i's calculation because we chose different ODE solvers, but it does not affect the final result as both algorithms reached the convergence quickly.

And the right-side three pictures show the measured states, the optimal discharges (inputs) and total active power (output) performances under the ROC control with a prediction horizon $H_p = 24$ hours. The predictive controller u^* is determined at sampling interval $[t_i, t_i + t_s) \rightarrow R^n$, by solving a free-endpoint open-loop optimal control problem over a prediction horizon $[t_i, t_i + H_p]$ (the index i is incremented with one unit in each iteration) with the constraint requirements fulfilled. Then the whole cycle of output

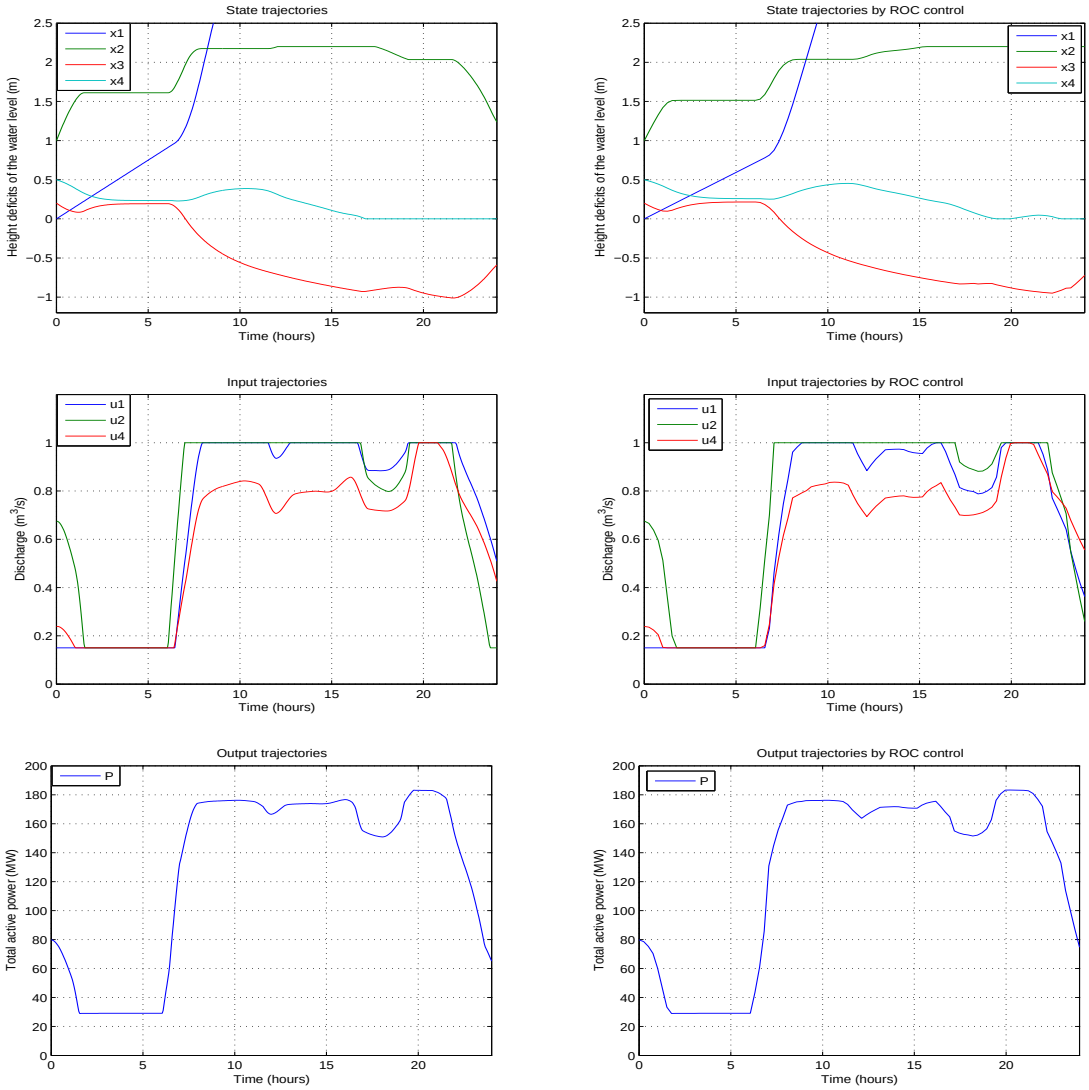


Figure 5.5: ROC Optimal control solution of Case 1

measurement, prediction, and input trajectory determination is repeated one sampling interval t_s forward, and the prediction horizon H_p moves also one t_s time-step ahead. Then a new input trajectory is applied at next sampling interval $[t_{i+1}, t_{i+1} + t_s)$, by solving a free-endpoint open-loop optimal control problem over a prediction horizon $[t_{i+1}, t_{i+1}+H_p]$, and a new system output is obtained, the prediction horizon is moved over the horizon $[t_{i+1}, t_{i+1} + H_p]$. The entire procedure is repeated at subsequent control intervals in order to get an updated control sequence, and horizons forward-move one t_s time-step further. Since in this case, the TPBVP ODEs model has a free-endpoint formulation, the values of

the terminal states $x(t_f) = x(t_i+H_p)$ may vary in each open-loop optimal control problem.

The optimal control variables are stored on discrete-time instants first, and then approximated to the intermediate time values via *spline interpolation*. It can be seen clearly that the state constraints of x_2 and x_4 are not violated in both optimization solution and ROC control, the optimal discharge u_1, u_2, and u_4 are also saturated all the time.

Case 2: Same as Case 1, only the whole optimization interval is changed to one week, i.e. $T = 168$. Its open-loop optimization solution is presented in the left-side three pictures of Figure 5.6, and ROC optimal control is shown in the right-side three pictures of Figure 5.6. Prediction horizon is $H_p = 24$ hours. State and control constraints are not violated in both optimization solution and ROC control. The entire procedure is repeated at each subsequent control intervals in order to get an updated control sequence, and horizons forward-move one t_s time-step further.

Case 3: In this case, one of the most difficult boundary-values optimization problem formulations is concerned, the end-point of state is fixed, i.e. the initial states $x(t_0)$, terminal states $x(t_f)$ are fixed and V is given only for initial guess of the final values of adjoint states. The boundary conditions are set in [LO76] as:

$$x(t_0) = [0,\ 0.5,\ -0.2,\ 0.1]^T, \tag{5.17}$$

$$x(t_f) = [16,\ 0.5,\ -0.2,\ 0.1]^T, \tag{5.18}$$

$$V = [20000,\ 15000,\ 10000,\ 10000]^T \tag{5.19}$$

So both initial and final values of the states x are fixed but the final value of adjoint states $p(t_f)$ has to be calculated in every iteration by

$$p_{i+1}(t) = K_i(t)^{-1}\left[x(t_f)^T - r_{i+1}(t)\right] + p_i(t) \tag{5.20}$$

Juha and I had a lot of difficulties in this case because the initial guess V from [LO76] did not work both in our codes even though it had been done a successful calculations

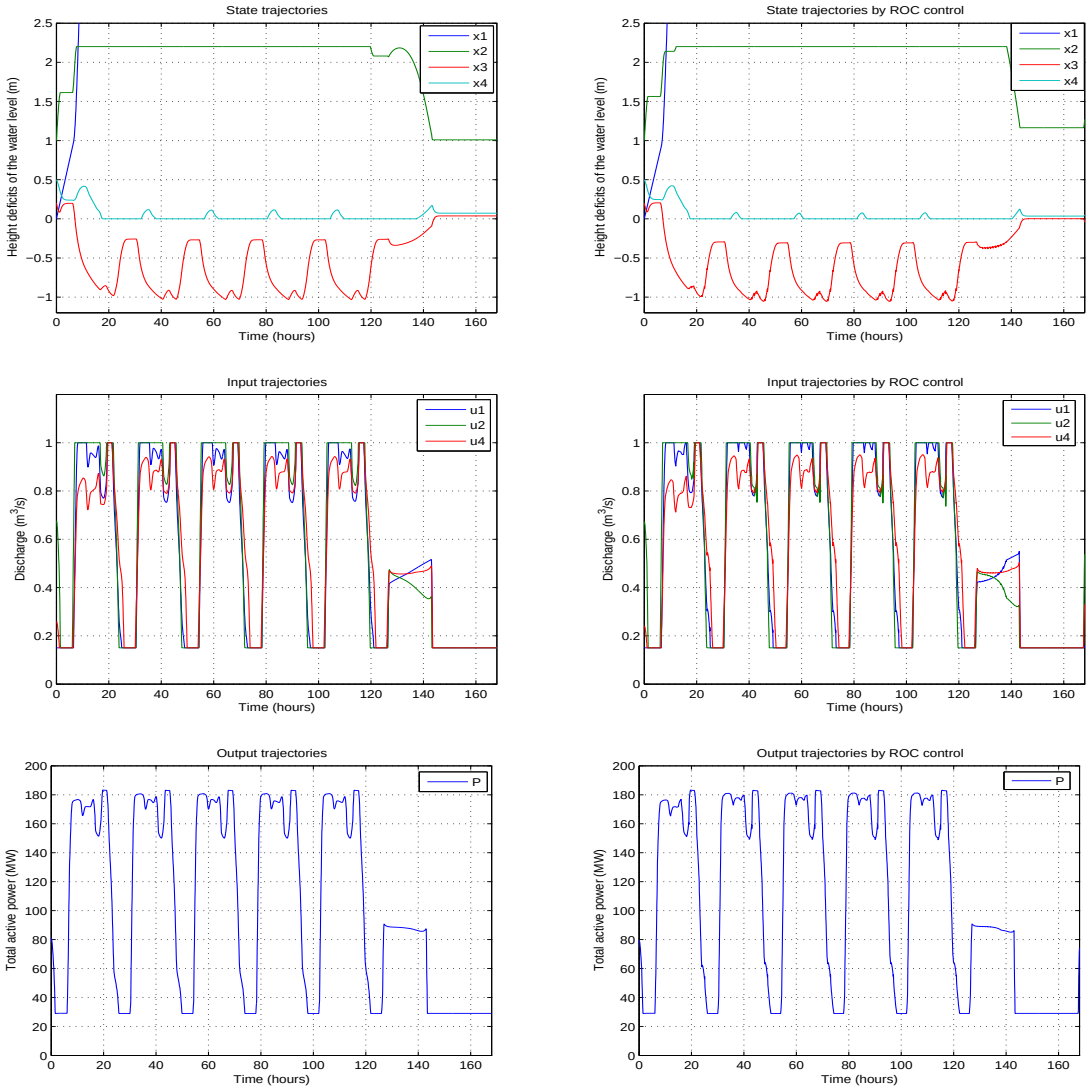


Figure 5.6: ROC Optimal control solution of Case 2

with this initial guess in [LO76]. We started with initial guess trials with lazier penalty term N at the beginning and later managed to raise it to be equal to 30. And Juha found another good initial guess to replace the original V with:

$$V = [20800,\ 15000,\ 10000,\ 10000]^T \tag{5.21}$$

Although only a tiny change has to be done, we can notice that how important it is to get the moderate guesses for the first initial states and adjoint states estimation in order

to solve such a complicated nonlinear hydro-power plant chain system optimal control problem, even with the aid of the back-and-forth shooting method, and again the rest of initial guesses, which are got from the measurement are always suitable for optimization solution according to the **Theorem 3.1** Belman's optimal principle [Bel57]. The other parameters are the same as the Case 1 has. The result is shown in Figure 5.7 when the norm of the error-array $\|p_{i+1} - p_i\|$ is smaller than 1e-5.

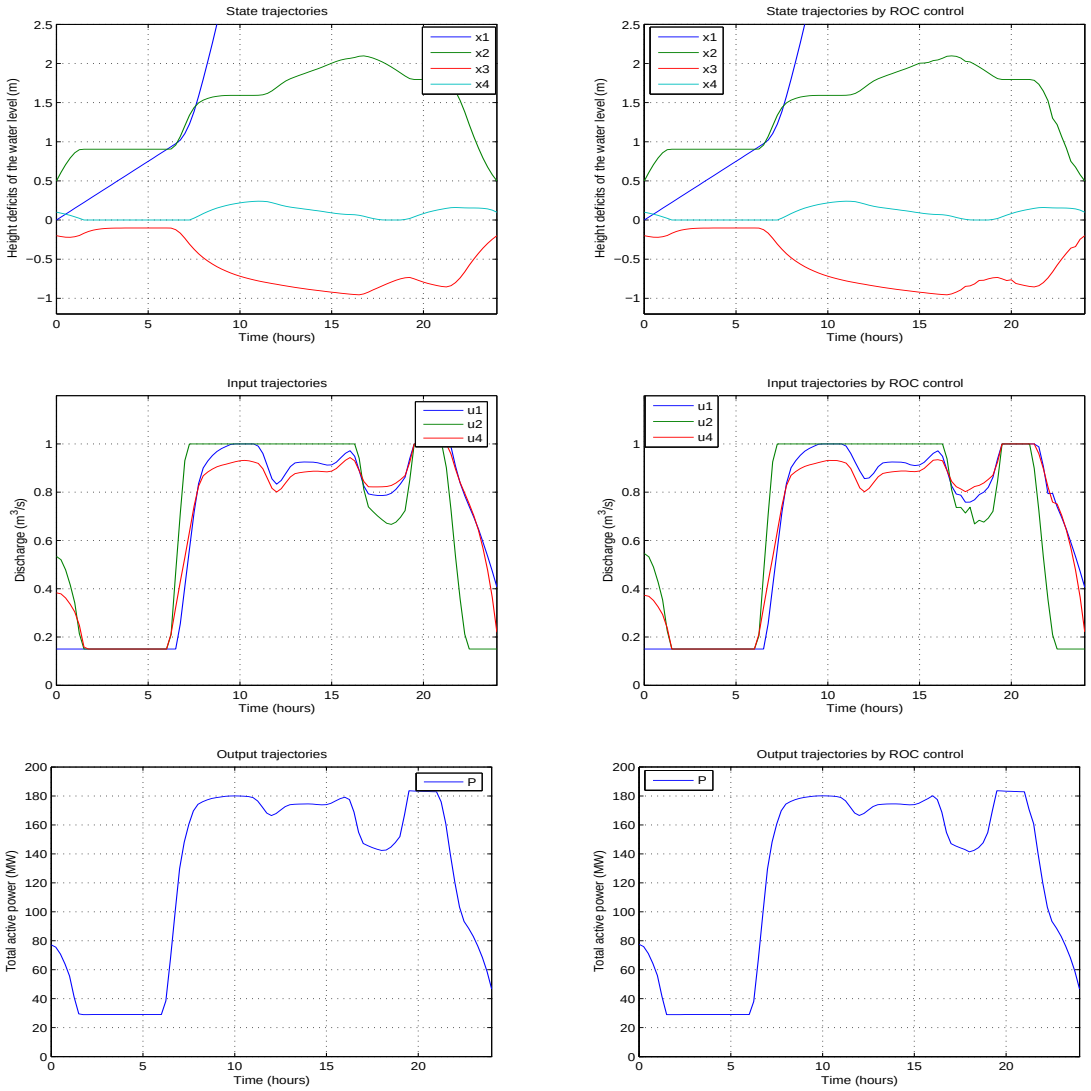


Figure 5.7: ROC Optimal control solution of Case 3

Its optimal solution is presented in the three left-side pictures of Figure 5.7. It demanded

10 iterations and the calculation time was about 2.317 seconds, the step size is set as 0.25. The results are same as Juha's but there are some differences between Figure 5.7 and figures plotted in [LO76] and it maybe caused because the initial guess for adjoint states p in [LO76] was chosen differently from Juha's and mine; in addition, the price curve $b(t)$ data was not given in [LO76], we have to manually measure it from the original figure, this also be the cause of some differences. And ROC control is shown in the right-side three pictures of Figure 5.7, prediction horizon is $H_p = 24$ hours. The state and control constraints did not violated in both optimization solutions and ROC control.

In this case, the predictive controller u^* is determined at sampling interval $[t_i, t_i + t_s) \rightarrow R^n$, by solving a fixed-endpoint open-loop optimal control problem (the final value of adjoint states $p(t_f)$ has to be calculated in every iteration) over a prediction horizon $[t_i, t_i + H_p]$ (the index i is incremented with one unit in each iteration) with the constraint requirements fulfilled. Then the whole cycle of output measurement, prediction, and input trajectory determination is repeated one sampling interval t_s forward, and the prediction horizon H_p moves also one t_s time-step ahead. Then a new input trajectory is applied at the next sampling interval $[t_{i+1}, t_{i+1} + t_s)$, by solving a free-endpoint open-loop optimal control problem over a prediction horizon $[t_{i+1}, t_{i+1} + H_p]$, and a new system output is obtained, the prediction horizon is moved over the horizon $[t_{i+1}, t_{i+1} + H_p]$. The entire procedure is repeated at subsequent control intervals in order to get an updated control sequence, and horizons forward-move one t_s time-step further. In addition, because the TPBVP ODEs model has a fixed-endpoint formulation, the values of the terminal states $x(t_f) = x(t_i + H_p)$ are always kept the same in each open-loop optimal control problem.

Case 4: It is also a fixed end point of state example, where the initial state $x(t_0)$, terminal state $x(t_f)$ are defined as:

$$x(t_0) = [0,\ 0.5,\ -0.2,\ 0.1]^T, \tag{5.22}$$

$$x(t_f) = [13,\ 0.5,\ -0.2,\ 0.1]^T, \tag{5.23}$$

In this case, the required initial guess for adjoint states p were not given in [LO76]. It was not that easy to find an good initial guess for it, the lazier penalty term N has to be considered, i.e., started from $N = 2$ and and tried to raise the value of N till 30. My colleague Juha firstly managed to find one suitable initial guess as

$$p(t) = [20700,\ 13300,\ 12700,\ 12660]^T \tag{5.24}$$

and I managed to find some other possiblities like

$$p(t) = [20800,\ 15000,\ 12500,\ 12500]^T \tag{5.25}$$

So both initial and final values of the states x are fixed also in this case and the final value of adjoint states $p(t_f)$ has to be also calculated in every iteration by

$$p_{i+1}(t) = K_i(t)^{-1}\left[x(t_f)^T - r_{i+1}(t)\right] + p_i(t) \tag{5.26}$$

constraints of inputs are:

$$\begin{aligned} u_{imin} &= 0.15, \quad i = 1, 2, 4 \\ u_{imax} &= 1, \quad i = 1, 2, \quad u_{4max} = 0.67, \end{aligned} \tag{5.27}$$

Other parameters are the same as in Case 1. The result is shown in Figure 5.8 when the norm of the error-array $\|p_{i+1} - p_i\|$ is smaller than 1e-5.

The optimal solution is presented in the left-side three pictures of Figure 5.8, it demanded 10 iteration turns and calculation time was about 2.292 seconds. Step size is set as 0.25. The results are the same as Juha got and also almost the same as the figures plotted in [LO76] with some small differences existing and it maybe caused because the initial guess for adjoint states p in [LO76] was chosen differently from Juha's and mine; in addition, the price curve $b(t)$ data was not given in [LO76], we have to manually measure it from the original figure, it maybe also cause some differences.

ROC control is shown in the right-side three pictures of Figure 5.8, prediction horizon is $H_p = 24$ hours. It can be easily seen that because TPBVP ODEs model also has a fixed-endpoint formulation, the values of the terminal states $x(t_f) = x(t_i + H_p)$ are kept the same in each open-loop optimal control problem as well. The states and control constraints of x_2 and x_4 are not violated in both optimization solution and ROC control, and the optimal discharge u_1, u_2, and u_4 are also saturated all the time.

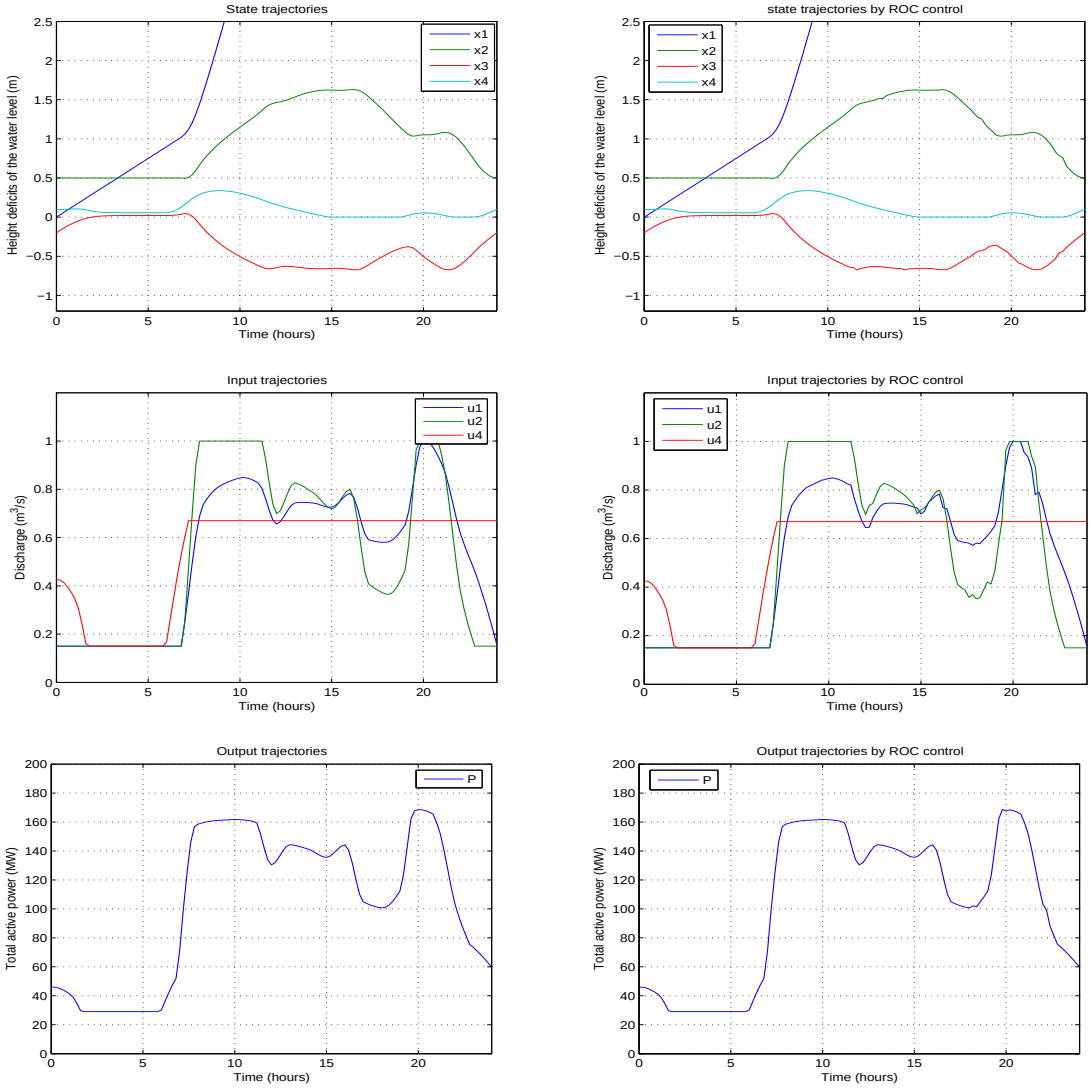


Figure 5.8: ROC Optimal control solution of Case 4

5.1.6 Remarks

The prediction $H_p = 24$ hours is used in all cases of this application, since the prediction with nonlinear TPBVP model requires to solve approximately the consecutive initial-value ODE problems, so the prediction horizon is chosen of the same length as the minimum optimization interval (24 hours). In Case 1, we solved the simplest situation that the initial value of state $x(t_0)$ is fixed without a fixed value of terminal state $x(t_f)$ at the end, so the fixed terminal adjoint state $p(t_f)$ is provided. In Case 2 we extended the simplest situation's optimization interval to one week, which only intends to exemplify the weekly optimization. The two cases because the TPBVP ODEs model has a free-endpoint formulation, the value of the terminal states $x(t_f) = x(t_i + H_p)$ may vary in each open-loop optimal control problem. In Case 3, we start to test the ROC algorithm with one of the most difficult TPBVP optimization problems, especially in this kind of constrained cased, where both the initial state $x(t_0)$ and the terminal state $x(t_f)$ are fixed, an adjoint state $p(t)$ is given for initial guess, which can vary in the whole optimization interval. The Case 4 exemplifies a case where one of the three turbine-generators of the Plant 4 is out of use. In these two cases, since the TPBVP ODEs model has a fixed-endpoint formulation, the values of the terminal states $x(t_f) = x(t_i + H_p)$ are kept the same in each open-loop optimal control problem.

It can be easily noticed from the above figures that the state constraints of x_2 and x_4 are not violated in both optimization solutions and ROC control, and thus the contribution of the penalty terms to the cost function is feasible. The optimal discharges u_1, u_2, and u_4 seen in every figure, are saturated all the time. The whole computations were performed on the CPU 2.8GHz and the algorithm and all application examples are implemented in the MATLAB environment. The differential equations are integrated by using fixed-step fourth-order Runge-Kutta method (ODE4) [ODE] with fixed step size 0.0625. The norm of the error-array $\|p_{i+1} - p_i\|$, set the same in every case, is smaller than $1e-5$. The convergence speed is quite fast and normally only few iterations are needed to achieve the solution of the problems, and the average computation time is normally around 2.275

seconds for back-and-forth shooting solution. The computation time of ROC control is roughly proportional to the number of repetitions (or the length of prediction horizon) of "inner loop" from the back-and-forth shooting solution. Hence, in a $H_p = 24$ hours ROC optimization control, the computation time normally requires about 63 seconds.

The ROC control seems to be a very effective control method for handling specially complex dynamic nonlinear system, and back-and-forth shooting method is also very powerful for the solution of TPBVP-form reduced from such a kind of optimization problem in spite of steep nonlinear penalty functions. The convergence of the algorithm is fast, and the speed of the program can still be increased considerably by compiling MEX files for further research. The results prove it a very promising algorithm in handling both controlling and the optimal solutions.

5.2 Exothermic Chemical Reactor

A MIMO-model exothermic chemical reactor is introduced in [LH93]. The process to be controlled is presented in Figure 5.9.

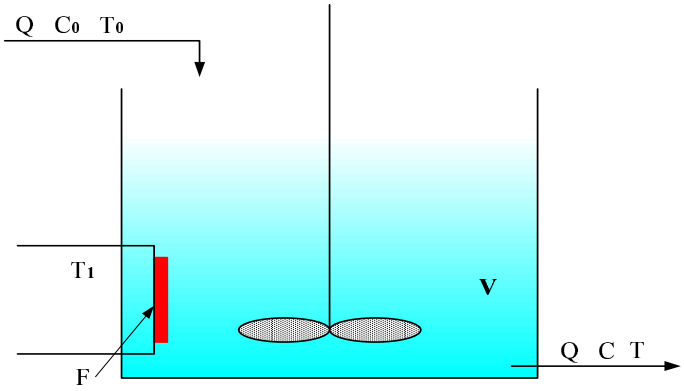


Figure 5.9: Exothermic chemical reactor

The mass M in the reactor and flow Q through the reactor are constants. The surface temperature of the heat exchanger T_1 is one controlled variable and the conductance F is constant. The temperature of the input flow T_0 is constant and the concentration of the substance A (C_0) is the second controlled variable. The reactor is an ideal mixer where

the temperature T and the concentration C are varying and the volume V is constant. The reaction produces energy ($\Delta H > 0$) and the balance equation was given by [LH93] as:

$$\begin{aligned} M\dot{C} &= Q(C_0 - C) - MCke^{E/RT} \\ c_n M\dot{T} &= \Delta H MCke^{E/RT} - c_n Q(T - T_0) - F(T - T_1) \end{aligned} \tag{5.28}$$

where the specific heat of the material is c_n and k, E and R are thermo-dynamical constants, and the numerical values of the parameters are defined as

$$\begin{aligned} M &= 10\ ton, \quad k = 0.002\ 1/s, \quad E/R = 3000\ ^oK, \\ c_n &= 0.25\ kcal/^oKkg, \quad \Delta H = 2500\ kcal/kg, \\ F &= 0.1 \ldots 1.0\ kcal/^oKs \end{aligned} \tag{5.29}$$

For convenience the equation was scaled in a way that all variables become dimensionless and the equation becomes as simple as possible in [LH93], and then the dynamical equation was reformed as follows:

$$\begin{aligned} \dot{c} &= q(c_0 - c) - ce^{-1/\Theta} \\ \dot{\Theta} &= ce^{-1/\Theta} - q(\Theta - \Theta_0) - f(\Theta - \Theta_1) \end{aligned} \tag{5.30}$$

and the numerical values of the parameters are defined like [LH93]

$$c_0 = 1.4, \quad q = 0.07, \quad f = 0.13, \quad \Theta_0 = 0.1, \quad \Theta_1 = 0.023 \tag{5.31}$$

The output reference trajectories are defined as constant to $y_{ref1}(t) = 1.0$ and $y_{ref2}(t) = 0.6$ for the simulation with the initial time $t_0 = 0$ seconds and end time $t_f = 5$ seconds.

In ROC control, the prediction horizon (and control horizon) is chosen as $H_p = 5$ seconds (default $H_c = H_p$). The weight factors of outputs and controls are defined as:

$$Q_{roc} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad R_{roc} = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix} \tag{5.32}$$

The cost function is in continuous-time as:

$$J_{roc} = \int_{t_0}^{t_f} \Big[(y - y_{ref})^T Q_{roc} (y - y_{ref}) + u^T R_{roc} u \Big] dt \tag{5.33}$$

Note that in true reality the weight of control derivatives (4-dimensional) will be taken into account, however in this academy experiment case, we just simply choose the weight of control to be concerned in the cost function.

In order to compare the result with ROC under the same condition, for general MPC control, the same prediction horizon and control horizon are chosen, i.e. $H_p = H_c = 5$ seconds. The output and control weights are

$$Q_{mpc} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad R_{mpc} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.34}$$

and its cost function based on discrete-time is formed as:

$$J_{mpc} = \sum_{t_0}^{t_f} \Big[(y - y_{ref})^T Q_{mpc} (y - y_{ref}) + u^T R_{mpc} u \Big] \tag{5.35}$$

Applying the ROC and the general MPC to the system with output tracking reference trajectories $y_{ref}(t) = [1;\ 0.6]$, sampling time is $T_s = 0.1$ seconds for both ROC and general MPC, and results are illustrated in Figure 5.10.

It can be seen clearly that both approaches can solve the "nearly" equivalent problem, however ROC performs much precisely approaching to the target, but general MPC, although it responds also rapidly at the start, its activities to reach the set-points seem much more sluggish. In addition, the system behaves smoother by ROC control than general MPC control in continuous-time. Furthermore, the start of ROC optimal control trajectory shows higher than MPC control trajectory because of its control weight setting, as comparing to $R_{mpc} = 0$, $R_{roc} = 0.001$ is still a big enough value. For illustration purpose, we change the control weight with one example to $R_{roc} = 0.01$, and the other

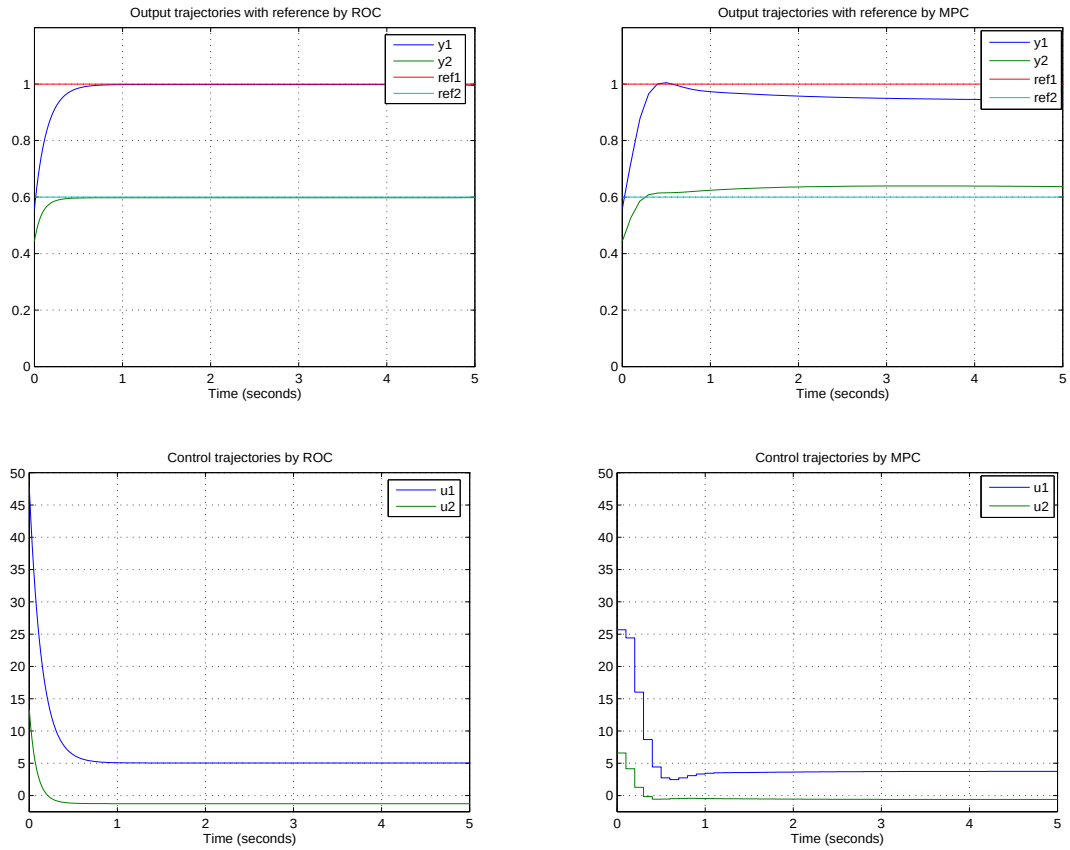


Figure 5.10: Exothermic chemical reactor controlled by ROC and general MPC

case to $R_{roc} = 0.0001$, the different control performance results are shown in Figure 5.11.

It can be clearly seen that when the weight of control in ROC is chosen to be some smaller value as $R_{roc} = 0.01$ (shown in left-side of the figure), the start of ROC optimal control trajectory shows smoother but the system yields more sluggish output response, and there is some error between system output and setpoint, however when the weight of control in ROC is $R_{roc} = 0.0001$ (shown in rght-side of the figure), opposite effect is given.

The all computations were performed on the CPU 2.8GHz and the application is implemented in the MATLAB environment. The differential equations are integrated by using a Runge-Kutta method (ODE45). The norm of the error-array $\|p_{i+1} - p_i\|$, is smaller than $1e - 5$. The convergence speed is very fast with only 3 iterations are needed to

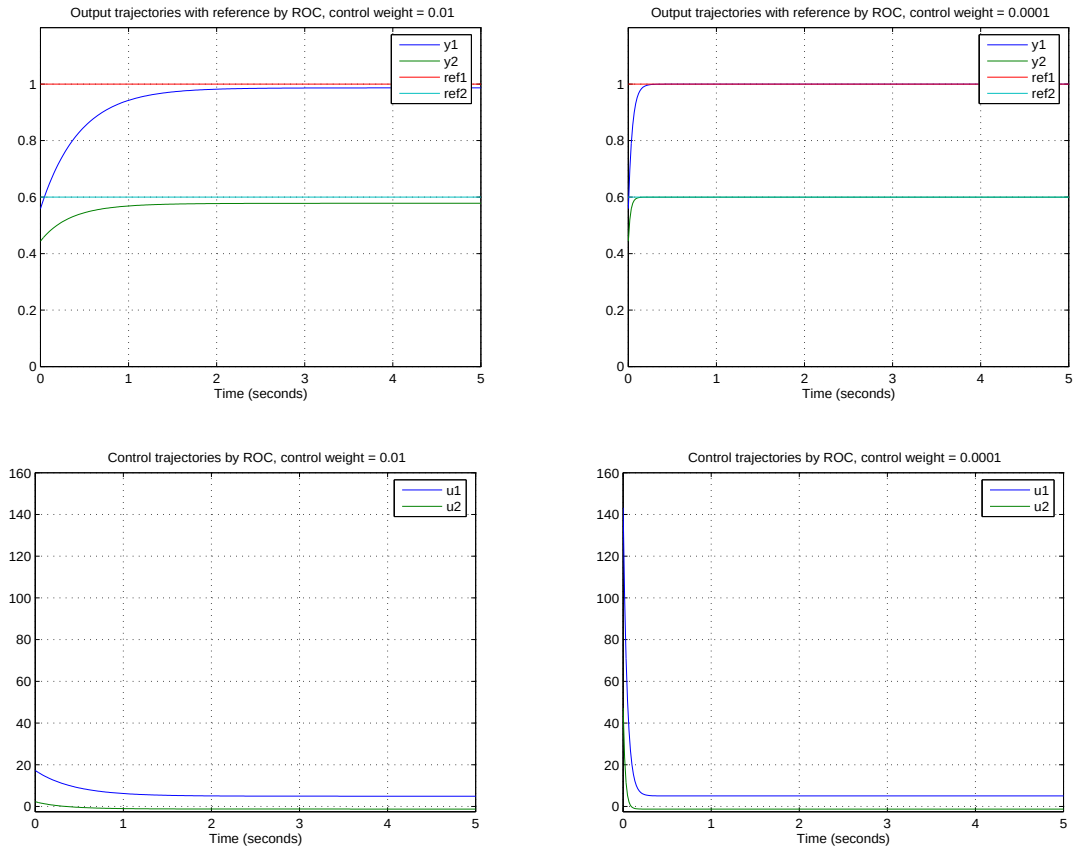


Figure 5.11: Exothermic chemical reactor controlled by ROC, control weight changing

achieve the solution of the problems, and the computation time is 1.611 seconds for back-and-forth shooting solution. The computation time of ROC control requires about 9 seconds.

Extending the states to contain a disturbance model, the covariance tuning parameters used are $Q_{kal} = 0.01$, and $R_{kal} = 0.01$. The initial state of the plant is $x_0 = [0,\ 0]$, and of the estimator $\hat{x}(k|k) = [0,\ 0]$. Then the optimal input trajectory and output trajectory turn to be as in Figure 5.12. The input disturbances are zero-mean white Gaussian distributed noise with variance equals to 0.05, and output disturbances are zero-mean white Gaussian distributed noise with variance equals to 0.1. Note that the linearized pair (C, A) describing the overall state-space realization of the combination of plant and disturbance models is observable or detectable for the state estimation design to succeed.

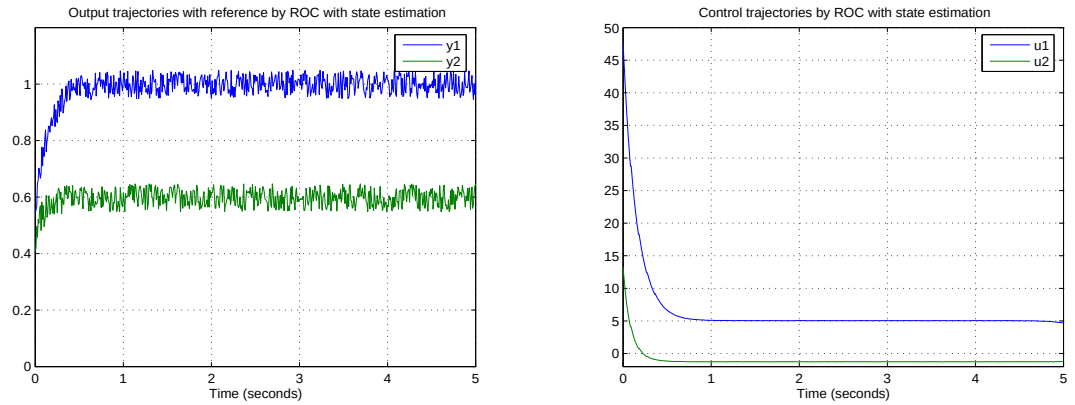


Figure 5.12: Exothermic chemical reactor controlled by ROC with state estimation

The model has been converted to discrete-time state-space form.

Concluding Remarks: The application studies in this chapter are firstly to testify the powerful capability of the ROC algorithm for dealing with fairly difficult and complex optimal control problems and at the same time explicitly handling multiple constrained states and inputs; and then a more practical industrial application is given as a nonlinear MIMO model. The control performance of ROC algorithm is evaluated by illustrative comparison with general MPC. All the results prove that ROC algorithm is a fairly promising algorithm in handling both controlling and the optimal solutions in continuous-time NMPC control field.

Chapter 6

Conclusions

This chapter summarizes the thesis work and also outlines some of the future developments this research suggests.

6.1 Conclusions

This thesis presents an efficient semi-analytical method, called repetitive optimal open-loop control (ROC), which is based on the model predictive control (MPC) strategy and allows explicit dealing with the optimization of constrained nonlinear processes in continuous-time. The whole structure of algorithm is built with an "inner loop" of TP-BVP optimization solution solved by the back-and-forth shooting method and an "outer loop" optimal control the system handled by ROC controller. The ROC algorithm and all applications are implemented in the MATLAB environment.

Discussion about handling nonlinear dynamic system with state and input constraints using the ROC algorithm of optimization is shown. The performance and theoretical properties of the ROC algorithm have been investigated and some attractive features are described as follows:

- Based on the framework of MPC in the general lines of the discrete-time, ROC algorithm extends a derivation to continuous-time field.

- In general NMPC, the considered nonlinear optimal control problem is normally non-convex, and an analytical solution is very difficult to find, instead a numerical optimization solution has to be generated. However, by using an open-loop feedback law, ROC algorithm builds an approximately analytical, or we say a semi-analytical solution between the optimal control variables and states. The resulting optimal control trajectory is well defined in a "continuously" varying sequence.

- ROC algorithm has a fairly powerful capacity in handling the practical dynamic systems with large variability and nonlinearity in practice with constraints.

The recommended back-and-forth shooting method for solving the reduced TPBVP optimization problem with the states and inputs constraints is included. This kind of TPBVP has a special form that the initial state is known, and half of the differential equations about state have fixed boundary conditions at the starting point. However if the terminal state is not fixed, then the other half of the differential equations about adjoint state has to be fixed at the endpoint. The differences and similarities between MPC and ROC have been also compared.

The study also extends to ROC control with state estimation (disturbance estimation). When perfect measurements of the current states are not available, an estimation of the initial state has to be done in the optimal control problem. An output feedback control, where a state estimator is used to obtain the current state from the output measurements, was constructed. Some zero-mean white Gaussian distributed noises are introduced to both controller and output measurements with linear and nonlinear dynamic systems using state estimation via Kalman filtering. Some numerical experiments are given for clear illustration.

Application studies are implemented in the end the research work, where the ROC algorithm is firstly be testified as an efficient algorithm for dealing with fairly difficult and complex optimal control problems and at the same time explicitly handling multiple constrained states and inputs. And the other application is an industrial MIMO exother-

mic chemical reactor. Solution proposal for universal control problem including reference tracking, input/output disturbance compensation, control weight changing is given. It is clearly shown by the results ROC algorithm achieves a good controller performance for solving general industrial optimization problems on-line.

6.2 Future development

The research described here naturally leads to several open questions and suggests some future developments. Because of the use of nonlinear models, the optimal control problem concerned in the ROC algorithm is usually a non-convex nonlinear problem, the analytical solution of which is very challenging to get.

In addition, although the disturbance models used for state estimation in ROC algorithm are studied, but in implementation there is only white Gaussian noise inserted in applications issues are concerned in both linear and nonlinear numerical examples, the work of either extended Kalman filters/linearized Kalman filters or moving horizon estimation (MHE) with unmeasured load-type disturbance model for nonlinear models are remained for further research interest. Also, since the ROC algorithm and applications are implemented with the MATALB code-language, the speed of programming can still be increased considerably by compiling MEX files for future research work.

Bibliography

[ABQ+99] F. Allögwer, T.A. Badgwell, J.S. Qin, Rawlings, J.B., and S.J. Wright. Nonlinear predictive control and moving horizon estimation - an introductory overview. In P.M.Frank, editor, *Advances in Control. Highlights of ECC'99*, pages 391–449. Springer, 1999.

[AF66] M. Athans and P.L. Falb. *Optimal Control*. McGraw Hill, New York, 1966.

[AMR95] U.M. Ascher, R.M.M. Mattheij, and R.D. Russell. Numerical solution of boundary value problems for ordinary differential equations. *Society for Industrial and Applied Mathematics (SIAM)*, 1995.

[BDLS99] H.G. Bock, M.M. Diehl, D.B. Leineweber, and J.P. Schlöder. Efficient direct multiple shooting in nonlinear model predictive control. In Keil, Mackens, Voss, and Werther, editors, *Scientific Computering in Chemical Engineering II: Simulation, Image Processing, Optimization, and Control*. Springer, 1999.

[BDLS00] H.G. Bock, M.M. Diehl, D.B. Leineweber, and J.P. Schlöder. A direct multiple shooting method for real-time optimization of nonlinear dae process. In F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control: series Progress in System Theory*, pages 245–267. Birkhuser, 2000.

[Bel57] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, 1957.

[Beq91] B.W. Bequette. Nonlinear predictive control using multi-rate sampling. *Canadian Journal of Chemical Engineering*, 69:136–143, 1991.

[Ber95] D.P Bertsekas. *Dynamic Programming and Optimal Control: Volume I.* Athena Scientific, Belmont, Massachusetts, 1995.

[BH75] A.E. Bryson and Yu-Chi. Ho. *Applied Optimal Control: Optimization, Estimation, and Control.* John Wiley and Sons, Hemisphere Publishing Corporation, Washington, D.C., 1975.

[Bry96] A.E. Bryson. Optimal control - 1950 to 1985. *IEEE Control Systems Magazine*, 16(3):26–33, 1996.

[CA98] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1218, 1998.

[CB99] E.F. Camacho and C. Bordons. *Model Predictive Control.* Springer-Verlag Berlin, Heidelberg, New York, 1999.

[CBG90] S. Chen, S.A. Billings, and P.M. Grant. Nonlinear system identification using neural networks. *International Journal of Control*, 51(6):1191–1214, 1990.

[CBG05] W.S. Chen, B.R. Bakshi, and P.K. Goel. Bayesian estimation via sequential monte carlo sampling constrained dynamic systems. 2005.

[CMT87a] D.W Clarke, C. Mohtadi, and P.S. Tuffs. Generalized predictive control - part i. the basic algorithm. *Automatica*, 23(2):137–148, 1987.

[CMT87b] D.W Clarke, C. Mohtadi, and P.S. Tuffs. Generalized predictive control - part ii. extensions and interpretations. *Automatica*, 23(2):149–160, 1987.

[DAOP95] F.J. Doyle, Babatunde A., B.A. Ogunnaike, and R.K. Pearson. Nonlinear model based control using second order volterra models. *Automatica*, 31(5):697–714, 1995.

[DBS+02] M. Diehl, H.G. Bock, J.P. Schloeder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[DFB+04] M. Diehl, R. Findeisen, H.G. Bock, J.P. Schlöder, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *IEE Proceedings - Control Theory Application*, 152(3):296–308, 2004.

[dNMS98] G. de Nicolao, L. Magni, and R. Scattolini. Stabilizing receding-horizon control of nonlinear time-varying systems. *IEEE Transactions on Automatic Control*, 47(3):1030–1036, 1998.

[dNMS00] G. de Nicolao, L. Magni, and R. Scattolini. Stability and robustness of nonlinear receding horizon control. In F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control*, pages 3–23. Birkhuser, 2000.

[dOB94] N.M.C de Oliveira and L.T. Biegler. Constraint handing and stability properties of model-predictive control. *AIChE Journal*, 40(7):1138–1155, 1994.

[dOB95] N.M.C de Oliveira and L.T. Biegler. An extension of newton-type algorithms for nonlinear process control. *Automatica*, 31(2):281–286, 1995.

[Eir83] T. Eirola. Convergence of the back-and-forth shooting method for solving two-point boundary-value problems. *Journal of Optimization Theory and Applications*, 41(4):559–572, 1983.

[Eir85] T. Eirola. *A Study of the Back-and-Forth Shooting Method*. PhD thesis, Helsinki University of Technology, 1985.

[ER90] J.W. Eaton and J.B. Rawlings. Feedback control of nonlinear processes using on-line optimization techniques. *Computers and Chemical Engineering*, 14(4-5):469–479, 1990.

[FA02] R. Findeisen and F. Allögwer. An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control*, Veldhoven, 2002.

[FJS95] B.A. Foss, T.A. Johansen, and A.V. Sorensen. Nonlinear predictive control using local models - applied to a batch fermentation process. *Control Engineering Practice*, 3(3):389, 1995.

[FPM97] K.P. Fruzzetti, A. Palazoglu, and K.A. MacDonald. Nonlinear model predictive control using hammerstein models. *Journal Process Control*, 7(1):31–41, 1997.

[FQ97] B.A. Foss and S.J. Qin. Interpolating optimizing process control. *Journal Process Control*, 7(1):129–138, 1997.

[GM86] C.E. Garcia and A.M. Morshedi. Quadratic programming solution of dynamic matrix control. *Chemical Engineering Communications*, 46:73–87, 1986.

[Hen96] J. Henttonen. *Receding Horizon Control Approach to Multivariable Systems*. PhD thesis, Tampere University of Technology, 1996.

[Hen98] M.A. Henson. Nonlinear model predictive control: Current status and future directions. *Computers and Chemical Engineering*, 23(2):187–202, 1998.

[HR05] E.L. Haseltine and J.B. Rawlings. Critical evaluation of extended kalman filtering and moving-horizon estimation. *Industrial and Engineering Chemistry Research*, 44(8):2451–2460, 2005.

[Jaz70] A.H. Jazwinski. *Stochastic Processes and Filtering theory*. Academic Press, New York, 1970.

[JNS01] Abonyi J., L. Nagy, and F. Szeifert. Fuzzy model-based predictive control by instantaneous linearization. *Fuzzy Sets and Systems*, 120(1):109–122, 2001.

[Kal60] R.E. Kalman. *Contributions to the Theory of Optimal Control*. Bull. Soc. Math., Mexicana, 1960.

[KB89] W.H. Kwon and D.G. Byun. Receding horizon tracking control as a predictive control and its stability properties. *International Journal of Control*, 50(5):1807–1824, 1989.

[KG88] 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(2):265–293, 1988.

[KM00] S.L.O. Kothare and M. Morari. Contractive model predictive control for constrained nonlinear systems. *IEEE Transaction on Automatic Control*, 45(6):1053–1071, 2000.

[KS72] H. Kwakernaak and R. Sivan. *Liner Optimal Control System*. Wiley, New York, 1972.

[Lau79] P.A.J. Lautala. On dynamic optimization of a hydro power plant chain. In *Link between Science and Applications of Automatic Control*. Pergramon Press, Oxford and New York, 1979.

[LB88] W.C. Li and L.T. Biegler. Process control strategies for constrained nonlinear systems. *Industrial and Engineering Chemistry Research*, 27(8):1421–1433, 1988.

[LB89] W.C. Li and L.T. Biegler. Multistep, newton-type control strategies for constrained, nonlinear process. *Chemical Engineering Research and Design*, 67:562–577, 1989.

[LH93] P.A.J. Lautala and J. Henttonen. Design and simulation of model predictive control in graphical environment. *Proceedings of the 12th IASTED International Conference in Modelling, Identification and Control*, pages 147–151, 1993.

[LKB98] G.P. Liu, V. Kadirkamanathan, and S.A. Billings. Predictive control for nonlinear systems using neural networks. *International Journal of Control*, 71(6):1119–1132, 1998.

[LKHB05] M. Lesken, L.B.M Kessel, P.M.J Hof, and O.H Bosgra. Nonlinear model predictive control with moving-horizon state and disturbance estimation, with application to msw combustion. In *Proceedings of 16th IFAC World Congress*, pages TU–A04–TP/12, Prague, 2005.

[LM67] J.H. Lee and L. Markus. *Foundation of Optimal Control Theory*. Wiley, New York, 1967.

[LO76] P.A.J. Lautala and P.J. Orava. On short-term optimal scheduling of hydroelectric power plant chains. In *Control Engineering Laboratory Report 2*. Helsinki University of Technology, 1976.

[LR94] J.H. Lee and N.L. Ricker. Extended kalman filter based nonlinear model predictive control. *Industrial and Engineering Chemistry Research*, 33(6):1530–1541, 1994.

[Lu95] P. Lu. Optimal predictive control of continuous nonlinear systems. *International Journal of Control*, 62(3):633–649, 1995.

[Mac02] J.M. Maciejowski. *Predictive Control: with Constraints*. Prentice-Hall, Englewood Cliffs, NJ, 2002.

[May79] P.S Maybeck. *Stochastic Models, Estimation and Control*. Academic Press, New York, 1979.

[May00] D.Q. Mayne. Nonlinear model predictive control. In F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control: series Progress in System Theory*, pages 24–44. Birkhuser, 2000.

[MDOP96] B.R. Maner, F.J.III Doyle, B.A. Ogunnaike, and R.K. Pearson. Nonlinear model predictive control of a simulated multivariable polymerization reactor using second-order volterra models. *Automatica*, 32(9):1285–1301, 1996.

[MHER95] E.S. Meadows, M.A. Henson, J.W. Eaton, and J.B. Rawlings. Receding horizon control and discontinuous state feedback stabilization. *International Journal of Control*, 62(6):1217–1229, 1995.

[ML99] M. Morari and J.H. Lee. Model predictive control: past, present and future. *Computers and Chemical Engineering*, 23(4-5):667–682, 1999.

[MM90] D.Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transaction on Automatic Control*, 35(7):814–824, 1990.

[MM93] H. Michalska and D.Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Transaction on Automatic Control*, 36(11):1623–1633, 1993.

[MR93] K.R. Muske and J.B Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, 1993.

[MR97] E.S. Meadows and J.B. Rawlings. Model predictive control. In M.A. Henson and D.E. Seborg, editors, *Nonlinear Process Control*, chapter 5, pages 233–310. Prentice Hall, Englewood Cliffs, NJ, 1997.

[MRRS00] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: stability and optimality. *Automatica*, 36(6):789–814, 2000.

[ODE] ODE4. Fixed-step fourth-order runge-kutta method (ode4), http://www.mathworks.com/support/tech-notes/1500/1510.html.

[ÖKG00] L. Özkan, M. V. Kothare, and C. Georgakis. Model predictive control of nonlinear systems using piecewise linear models. *Computers and Chemical Engineering*, 24(2-7):793–799, 2000.

[OL76] P.J. Orava and P.A.J. Lautala. Back-and-forth shooting method for solving two-point boundary-value problems. *Journal of Optimization Theory and Applications*, 18(4):485–497, 1976.

[PBGM62] L.S. Pontryagin, V.G. Boltyanskii, R.V. Gamkrelidze, and E.F. Mishchenko. *The Mathematical Theory of Optimal Processes*. Wiley, New York, 1962.

[PK94] T. Proll and M.N. Karim. Model-predictive ph control using real-time narx approach. *AIChE Journal*, 40(2):269–282, 1994.

[POD96] R.K. Pearson, B.A. Ogunnaike, and F.J. Doyle. Identification if structure constrained second-order volterra models. *IEEE Transaction Acoustics, Speech and Signal Processing*, 44:2837–2846, 1996.

[Pro63] A. I. Propoi. Use of lp methods for synthesizing sampled-data automatic systems. *Automation and Remote Control*, 1963.

[QB96] S.J. Qin and T.A. Badgwell. An overview of industrial model predictive control technology. In *Fifth International Conference on Chemical Process Control*, pages 232–256, Lake Tahoe, CA, 1996.

[QB00] S.J. Qin and T.A. Badgwell. An overview of nonlinear model predictive control applications. In F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control*, pages 369–392. Birkhuser, 2000.

[Rao00] C.V. Rao. *Moving Horizon Strategies for the Constrained Monitoring and Control of Nonlinear Discrete-Time Systems*. PhD thesis, University of Wisconsin-Madison, 2000.

[Raw99] J.B. Rawlings. Tutorial: Model predictive control. In *Amercian Control Conference Proceedings*, San Diego, California, 1999.

[RLR96] D.G. Robertson, J.H. Lee, and J.B. Rawlings. A moving horizon-based approach for least-squares estimation. *AICHE*, 42(8):2209–2224, 1996.

[RMM93] J.B. Rawlings, E.S. Meadows, and K.R. Muske. Nonlinear model predictive control: a tutorial and survey. In *ADCHEM' 94 Proceeding*, Kyoto, Japan, 1993.

[Ros03] J.A. Rossiter. *Model-Based Predictive Control: A practical approach*. CRC Press, Boca Raton, Florida, 2003.

[RRM03] C.V. Rao, J.B. Rawlings, and D.Q. Mayne. Constrained state estimation for nonlinear discrete-time systems: stability and moving horizon approximations. *IEEE Transaction on Automatic Control*, 48(2):246–258, 2003.

[RRTP78] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: applications to industrial processes. *Automatica*, 14:413–428, 1978.

[RSR95] Hartl R.F., Sethi S.P., and Vickson R.G. Maximum principles for optimal control problems with state constraints. *SIAM*, 37(2):181218, 1995.

[RVD03a] Holsapple R., R. Venkataraman, and D. Doman. A modified simple shooting method for solving two-point boundary-value problems. *Aerospace Conference, IEEE Proceedings*, 6(5):2783–2790, 2003.

[RVD03b] Holsapple R., R. Venkataraman, and D. Doman. A new, fast numerical method for solving two-point boundary-value problems. *Journal of Guidance, Control and Dynamics*, 2003.

[SA97] G.R. Sriniwas and Y. Arkun. A global solution to the nonlinear model predictive control algorithms using polynomial arx models. *Computers and Chemical Engineering*, 21(4):431–439, 1997.

[SA98] B. Schenker and M. Agarwal. Predictive control of a bench-scale chemical reactor based on neural-network models. *IEEE Transaction on Automatic Control*, 6(3):388–400, 1998.

[SJ98] T.S. Schei and T.A. Johansen. Nonlinear model based/model predictive control with constraints and with/without nonlinear observer. SINTEF Electronics and Cybernetics, Trondheim, Norway, 1998.

[SM97] H.T. Su and T.J. McAvoy. Artificial neural networks for nonlinear process identification and control. In Henson M.A. and D.E. Seborg, editors, *Nonlinear Process Control*, pages 371–428. Prentice-Hall, 1997.

[SW97] H.J. Sussmann and J.C. Willems. 300 years of optimal control: from the brachystochrone to the maximum principle. *IEEE Control Systems Magazine*, 17(3):32–44, 1997.

[TR02] M.J. Tenny and J.B. Rawlings. Efficient moving horizon estimation and nonlinear model predictive control. In *Proceedings of the American Control conference*, pages 4475–4480, Anchorage, 2002.

[TS85] T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modeling and control. *IEEE Transaction On System Man and Cybernetics*, SMC-15:116–132, 1985.

[WMM+00] M.J. Willis, G.A. Montague, C.D. Massimo, M.T. Tham, and A.J. Morris. Artificial neural networks in process estimation and control. *Automatica*, 28(6):1181–1187, 2000.

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O. Box 527
FIN-33101 Tampere, Finland