Neurodynamic Approaches to Model Predictive Control

PAN, Yunpeng

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Mechanical and Automation Engineering

The Chinese University of Hong Kong
August 2009

Thesis Assessment Committee

Professor Yam, Yeung (Chair)

Professor Wang, Jun (Thesis Supervisor)

Professor Liu, Yunhui (Committee Member)

Professor Liao, Lizhi (External Examiner)

Abstract

Model predictive control (MPC), also known as receding horizon control (RHC), is a powerful technique for optimizing the performance of control systems. However, the high computational demand in solving optimization problem associated with MPC in real-time is a major obstacle. In addition, when process models are unknown, nonlinear or contain uncertainties, there would be more challenges in the analysis and synthesis of nonlinear and robust MPC systems. To overcome these obstacles and challenges, recurrent neural networks (RNNs) are employed in controller design. This thesis is concentrated on the analysis and synthesis of RNN-based MPC for linear systems, nonlinear systems, and systems with uncertainties.

This thesis consists of seven chapters. We first develop RNN-based MPC schemes for linear systems. As linear MPC problems can be generally formulated as linear programming (LP) or quadratic programming (QP) problems, two RNNs are applied in controller design to solve the associated LP or QP problems. Both RNNs have desired convergence property and relatively lower computational complexity. For nonlinear affine systems, general nonlinear systems, and unknown systems, MPC based on RNNs are also developed. When the system model is unknown, two types of RNNs (e.g., echo state network and simplified dual network) are employed for system identification and optimization, respectively. Furthermore, the RNN-based nonlinear MPC is applied to mobile robot tracking. When bounded uncertainties are considered in linear systems, the MPC synthesis problem can be formulated as a minimax optimization problem, a discrete-time RNN is developed for solving this minimax problem. The proposed RNN has global ex-

ponential convergence property and can be implemented using digital hardware. Simulation results are provided to demonstrate the effectiveness and efficiency of the proposed RNN-based approaches.

摘要

模型預測控制，又名滾動時域控制，是一種用於優化控制系統的技術。然而，用於解決模型預測控制中優化問題的高計算複雜度成為即時應用中的障礙。 另外，當模型是未知，非線性，或者含有不確定因素時，設計控制器會更加困難。為了克服這些困難，回饋神經網路被應用於控制器設計。 此碩士論文的內容主要為針對線性，非線性，以及含有不確定因素系統的基於回饋神經網路的模型預測控制器的分析與設計。

此篇論文包含七個章節。 我們首先提出了基於回饋神經網路的線性模型預測控制方法。線性模型預測控制問題可以被轉化為線性規劃或二次規劃問題，兩種回饋神經網路分別應用於解決模型預測控制中的線性規劃和非線性規劃問題，此二種神經網路均含有全局收斂性以及較低的計算複雜度。 對於非線性仿射系統，一般非線性系統以及未知系統，我們也提出了相應的基於回饋神經網路的模型預測控制方法。 當系統未知時，兩類回饋神經網路（回聲狀態神經網路以及簡化對偶神經網路）分別應用於系統辨識和優化。 另外，基於回饋神經網路的非線性模型預測控制方法被應用於移動機器人跟蹤控制。 當線性模型中含有不確定因素時，預測控制問題可轉化為最大最小優化問題，我們提出一個離散回饋神經網路來解決此問題，此神經網路被證明含有指數收斂性，並可以用數位硬體實現。仿真結果驗證了基於回饋神經網路的非線性模型預測控制方法的效果。

Acknowledgement

I would like to express my deep gratitude to my supervisor, Prof. Jun Wang, for his supervision throughout these three years. Without his continuous encouragement, kindness and help, this thesis is no way in its current form. His profound knowledge and new idea on neural networks are very helpful for my research.

Special thanks are also due to Prof. Zhigang Zeng for providing me with valuable comments on this project and presentation each time, which make this thesis better in many respects.

I would like to convey my appreciation to my officemates and friends Qingshan Liu, Shenshen Gu, Jin Hu, Xiaolin Hu and many other people without space to list for their support, help and friendship. In particular, I would like to express my thanks to my family. I should never be here without their unconditional love, encouragement and support.

This work is dedicated to my parents.

Contents

Chapter 1

Introduction

1.1 Model Predictive Control

In recent years, the requirements for the quality of industrial process control increased significantly, due to the increased complexity of the process plants and sharper specifications of product quality. Among various kinds of industry process control techniques, model predictive control (MPC) is a promising one. Model predictive control (MPC), also known as receding horizon control (RHC), is an advanced control strategy for optimizing the performance of control systems. MPC generates control actions by optimizing an objective function repeatedly over a finite moving prediction horizon, within system constraints, and based on a model of the dynamic system to be controlled. As the most effective multivariable control technology, MPC has many desirable features suitable for industrial applications. One of the key advantages of MPC is its ability to deal with input and output constraints; another is that MPC can be naturally applied for multivariable process control. Because of these advantages, MPC has been used in numerous industrial applications in the refining, petrochemical, chemical, pulp, paper, and food processing industries. The development and applications of MPC technology can be traced back 30 years to late seventies, when the first

MPC strategy, which was based on quadratic programming (QP), was presented by Richalet [1]. Since then, the MPC research and development have grown significantly. Recent survey and review of MPC algorithms and technologies were introduced in [2][3].

Most control techniques do not consider the future implication of current control actions. MPC applies on-line optimization to a system model. By taking the current state as an initial state, a cost-minimization control strategy is computed at each sample time, and at the next computation time interval, the calculation repeated with a new state. The basic structure of MPC is shown in Fig. 1.1. As the process model of MPC is usually expressed with linear or quadratic criterion, MPC problems can be generally formulated as linear programming or quadratic programming problems. As a result, they can be solved using solution methods for linear and quadratic programming problems. A key issue for MPC synthesis lies in online optimization. The success of any MPC implementation depends on the effectiveness and efficiency of the solution method used. One possible and very promising approach to dynamic optimization is to apply recurrent neural networks (RNNs).

1.2 Neural Networks

In numerous of science and engineering applications, such as robot control, manufacturing system design, signal and image processing, and pattern recognition, the problems can be formulated as linear or nonlinear programming problems [4][5]. Over years, a variety of numerical algorithms have been developed for solving linear and nonlinear programming problems [5], such as the gradient projection method by Rosen in 1960 [6] and the penalty function method by Zangwill in 1967 [7]. However, in many engineering applications, real-time solutions are often needed. One promising approach for solving optimization problems on real time is to employ recurrent neural networks (RNNs) based on circuit implementa-

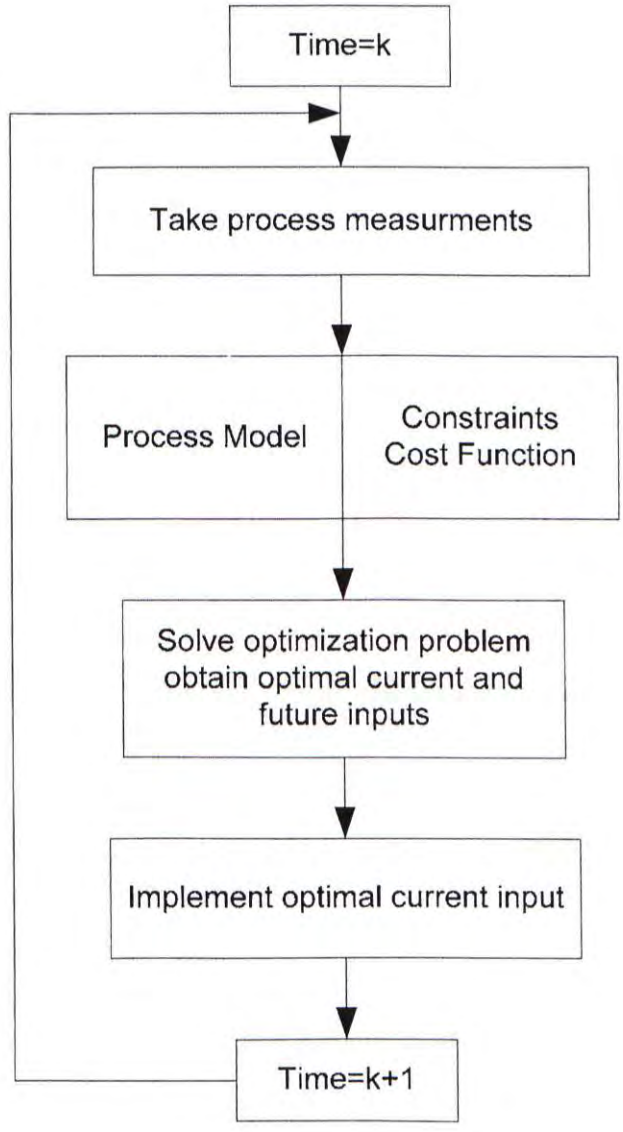

Figure 1.1: Basic structure of MPC algorithm.

tion. As the counterparts of biological neural systems, properly designed artificial neural networks can serve as goal-seeking computational models for solving various optimization problems in many applications [8]-[11].

Compared with traditional numerical methods for constrained optimization, neural network has several advantages: first, it can solve many optimization problems with time-varying parameters; second, it can handle large-scale problems with its parallelizable ability; third, it can be implemented effectively using VLSI and optical technologies [12]. Therefore, neural network can solve optimal control problems in running times at the orders of magnitude much faster than the most popular optimization algorithms executed on general-purpose digital computers. Application areas of neural networks include, but are not limited to, system modeling, mathematical programming, associative memory, combinatorial optimization, pattern recognition and classification, robotic and process control, and design and planning.

In 1940s, the first conceptual elements of neural networks were introduced. Since then, numerous neural network models have been developed. In the past two decades, recurrent neural networks for optimization and their engineering applications have been widely investigated. Tank and Hopfield proposed the first working recurrent neural network implemented on analog circuits [13], their work inspired many researchers to develop other neural networks for solving linear and nonlinear optimization problems.

Kennedy and Chua [14] presented a neural network with a finite penalty parameter for nonlinear programming which can converge to approximate optimal solutions. To avoid using the penalty parameter, many other methods have been introduced to develop various neural networks. Zhang and Constantinides [15] proposed the Lagrangian network based on the Lagrangian method which had a two-layers structure and this neural network was globally convergent to an optimal solution if only the objective function was strictly convex [16]. Wang [17][18] developed the deterministic annealing network for linear and nonlinear convex

programming. Xia [19] proposed some primal neural networks for solving convex quadratic programming problems. The primal-dual neural networks [20][21] with two-layers architecture were proposed for solving linear and nonlinear programming problems. The primal-dual neural networks based on the primal-dual method are globally convergent to the primal and dual solutions of the optimization problems. The primal-dual neural networks have been widely utilized to the assignment problems [22][23] and online resolving constrained kinematic redundancy in robot motion control [24]. In [25]-[27], the dual neural networks as simplified forms of the primal-dual neural networks were presented to solve convex quadratic programming problems utilizing only the dual variables. In order to simplify the architecture of the dual neural network, a simplified dual neural network was proposed for solving quadratic programming problems [28]. Based on the projection method [29], the projection neural network was proposed for solving general convex programming problems [30]-[36] which was globally convergent to exact optimal solutions. Recently, Forti, Nistri and Quincampoix proposed a generalized neural network for solving non-smooth nonlinear programming problems based on the gradient and penalty parameter methods [37]. The delayed neural networks were proposed for solving convex quadratic programming problems [38][39][40]. In [41][42], Liu and Wang proposed two one-layer recurrent neural networks for solving linear and quadratic programming problems. The one-layer recurrent neural networks, which number of neurons is equal to that of decision variables in the programming problems, have more simply architecture complexity than the other neural networks such as Lagrangian network and projection network.

1.3 Existing studies

Several studies on linear MPC based on neurodynamic optimization have been presented [43]-[50]. In [43], the Hopfield neural network was applied to linear

MPC. But the the control performance is compromised as the approximation strategy yields sub-optimal solutions. In [44], a discrete-time structured neural network was proposed to solve the QP problem involved in linear MPC exactly. However, it has a 3-layer structure which is complex for implementation. In [45], the dual neural network in [28] was adopted for multi-variable generalized predictive control. In [46], two RNNs with simple structure and global convergence property were applied in model predictive controller design. In [47], a robust MPC scheme based on a discrete-time RNN was proposed for linear systems with bounded uncertainties. These studies showed that neural networks have good performance in solving optimization problem associated with linear MPC. In recent years, nonlinear MPC using RNNs were also investigated. In [48], two recurrent radial basis function networks were applied for system identification and control, respectively. In [49], an RNN was adopted for prediction modeling. In [50], nonlinear MPC synthesis problem was reformulated to a QP problem and an RNN was used for solving the QP problem.

1.4 Thesis structure

In this thesis, MPC schemes based on RNNs are developed for linear systems, nonlinear systems, and systems with uncertainties. The thesis is divided into seven chapters organized as follow:

The first chapter gives a brief introduction to MPC and RNN, some existing related works are also introduced.

In Chapter 2, linear MPC synthesis problem is formulated as both linear and quadratic programming problems, two RNNs with global convergence property and low computational complexity are applied in controller design. A comparison was made between the two neural network approaches.

Chapter 3 presents a MPC scheme for nonlinear affine systems based on the simplified dual network. Simulation results show that the proposed approach is

effective and efficient compared with linear MPC.

In Chapter 4, the RNN-based nonlinear MPC scheme is proposed and applied to mobile robot tracking. The simplified dual network and recursive learning algorithm are employed in MPC design. Compared with linear MPC, the RNN-based approach gives better tracking result with less errors.

Chapter 5 presents a MPC scheme for unknown dynamic systems based on RNNs. Two types of RNNs (e.g., echo state network and simplified dual network) are employed for system identification and optimization, respectively. A recursive algorithm for RNN learning is developed, the proof of convergence property is given.

In Chapter 6, we formulate robust MPC as a quadratic minimax optimization problem. A discrete-time RNN for minimax optimization is developed and applied, whose global exponential convergence property is proved. Furthermore, a comparison is made between the proposed approach and linear matrix inequalities approach.

Chapter 7 concludes this thesis, describes some unsolved problems, and points out future research in this area.

□ **End of chapter.**

Chapter 2

Two Recurrent Neural Networks Approaches to Linear Model Predictive Control

2.1 Problem Formulation

Consider the following linear discrete-time system:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ y(k) &= Cx(k), \end{aligned} \tag{2.1}$$

with the constraints

$$\begin{aligned} u_{\min} \leq & u(k) \leq u_{\max}, \\ \Delta u_{\min} \leq & \Delta u(k) \leq \Delta u_{\max}, \\ y_{\min} \leq & y(k) \leq y_{\max}, \end{aligned} \tag{2.2}$$

which represents the dynamics of the plant under consideration. In (2.1)-(2.2), $k \geq 0$, $x(k) \in \Re^n$ is the state vector, $u(k) \in \Re^m$ is the input vector, and $y(k) \in \Re^p$ is the output vector. $u_{\min} \leq u_{\max}$ and $y_{\min} \leq y_{\max}$ are vectors of upper and lower bounds.

Model predictive control is a step-by-step optimization technique: at each sampling time, measure of estimate the current state, obtain the optimal input vector by solving the following optimization problem is solved at each time k.

2.1.1 Quadratic Programming Formulation

With a quadratic criterion, MPC can be formulated as the following optimization problem:

$$\begin{aligned}
\min \quad & \sum_{j=1}^{N}[r(k+j|k)-y(k+j|k)]^T Q[r(k+j|k)- \\
& y(k+j|k)] + \sum_{j=0}^{N_u-1} \Delta u(k+j|k)^T R \Delta u(k+j|k) \\
\text{s.t.} \quad & u_{\min} \leq u(k+j|k) \leq u_{\max}, \quad j=0,...,N_u-1; \\
& \Delta\, u_{\min} \leq \Delta u(k+j|k) \leq \Delta u_{\max}, \quad j=0,...,N_u-1; \\
& y_{\min} \leq y(k+j|k) \leq y_{\max}, \quad j=1,...,N;
\end{aligned} \tag{2.3}$$

where k is the current time step, $y(k+j|k)$ denotes the predicted output, $r(k+j|k)$ denotes the reference trajectory of output signal (desired output) at sampling instant k, and $\Delta u(k+j|k)$ denote the input increment, where $\Delta u(k+j|k) = u(k+j|k) - u(k-1+j|k)$. N is the predictive horizon, where $1 \leq N$. N_u denote the control horizon, where $0 < N_u \leq N$. After N_u control moves, $\Delta u(k+j|k)$ become zero. $Q \in \Re^{p\times p}$ and $R \in \Re^{m\times m}$ are appropriate weighting matrices.

According to the process model (2.1):

$$\begin{aligned}
y(k+j|k) = CA^j x(k) + C\sum_{i=0}^{j-1} A^i Bu(k+j-i-1|k), \\
j=1,...,N.
\end{aligned} \tag{2.4}$$

Define following vectors:

$$\bar{y}(k) = [y(k+1|k) \quad \cdots \quad y(k+N|k)]^T \in \Re^{Np},$$

$$\bar{u}(k) = [u(k|k) \quad \cdots \quad u(k+N_u-1|k)]^T \in \Re^{N_u m},$$

$$\Delta\bar{u}(k) = [\Delta u(k|k) \quad \cdots \quad \Delta u(k+N_u-1|k)]^T \in \Re^{N_u m},$$

$$\bar{r}(k) = [r(k+1|k) \quad \cdots \quad r(k+N|k)]^T \in \Re^{Np},$$

where the reference trajectory $\bar{r}(k)$ is known in advance. The predicted output $\bar{y}(k)$ are expressed in the following form:

$$\begin{aligned} \bar{y}(k) &= Sx(k) + M\bar{u}(k) \\ &= Sx(k) + M\Delta\bar{u}(k) + Vu(k-1), \end{aligned} \tag{2.5}$$

where

$$S = [CA \quad CA^2 \quad \cdots \quad CA^N]^T \in \Re^{Np\times n},$$

$$V = \begin{bmatrix} CB \\ C(A+I)B \\ \vdots \\ C(A^{N_u-1}+\cdots+A+I)B \\ C(A^{N_u}+\cdots+A+I)B \\ \vdots \\ C(A^{N-1}+\cdots+A+I)B \end{bmatrix} \in \Re^{Np\times m},$$

$$M = \begin{bmatrix} CB & \dots & 0 \\ C(A+I)B & \dots & 0 \\ \vdots & \ddots & \vdots \\ C(A^{N_u-1}+...I)B & \dots & CB \\ C(A^{N_u}+...I)B & \dots & C(A+I)B \\ \vdots & \ddots & \vdots \\ C(A^{N-1}+...I)B & \dots & C(A^{N-N_u}+...I)B \end{bmatrix},$$

$M \in \Re^{Np\times N_u m}$, I denotes the identity matrix. Define vectors:

$$\Delta\bar{u}_{\max} = [\Delta u_{\max} \cdots \Delta u_{\max}]^T \in \Re^{N_u m},$$

$$\bar{u}_{\min} = [u_{\min} \cdots u_{\min}]^T \in \Re^{N_u m},$$

$$\bar{u}_{\max} = [u_{\min} \cdots u_{\max}]^T \in \Re^{N_u m},$$

$$H = \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & I & I \end{bmatrix} \in \Re^{N_u m \times N_u m}.$$

Thus, the optimization problem (2.3) can be expressed in the following form:

$$\begin{aligned} \min \quad & [\bar{r}(k) - Sx(k) - M\Delta\bar{u}(k) - Vu(k-1)]^T Q[\bar{r}(k) - \\ & Sx(k) - M\Delta\bar{u}(k) - Vu(k-1)] + \Delta\bar{u}^T(k) R \Delta\bar{u}(k) \\ \text{s.t.} \quad & \bar{u}_{\min} \leq \bar{u}(k) + H\Delta\bar{u}(k) \leq \bar{u}_{\max} \\ & \Delta\bar{u}_{\min} \leq \Delta\bar{u}(k) \leq \Delta\bar{u}_{\max} \\ & \bar{y}_{\min} \leq \bar{y}(k) + M(k)\Delta\bar{u}(k) \leq \bar{y}_{\max} \end{aligned} \tag{2.6}$$

By defining the variable vector $v = \Delta\bar{u}(k) \in \Re^{N_u m}$, the problem (2.3) can be rewritten as a standard quadratic programming problem form:

$$\begin{aligned} \min \quad & \frac{1}{2} v^T W v + c^T v \\ \text{s.t.} \quad & l_{\min} \leq Gv \leq l_{\max} \end{aligned} \tag{2.7}$$

where the coefficient matrices and vectors are

$$l_{\min} = (-\infty \quad \Delta\bar{u}_{\min}]^T \in \Re^{3N_u m + 2Np},$$

$$l_{\max} = [b \quad \Delta\bar{u}_{\max}]^T \in \Re^{3N_u m + 2Np},$$

$$W = 2(M^T Q M + R) \in \Re^{N_u m \times N_u m},$$

$$c = -2M^T Q(\bar{r}(k) - Sx(k) - Vu(k-1)) \in \Re^{N_u m},$$

$$E = [-H \quad H \quad -M \quad M]^T \in \Re^{(2N_u m + 2Np) \times N_u m},$$

$$G = [E \quad I]^T \in \Re^{(3N_u m + 2Np) \times N_u m}$$

$$b = \begin{bmatrix} -\bar{u}_{\min} + \bar{u}(k) \\ \bar{u}_{\max} - \bar{u}(k) \\ -\bar{y}_{\min} + \bar{y}(k) \\ \bar{y}_{\max} - \bar{y}(k) \end{bmatrix} \in \Re^{2N_u m + 2Np}.$$

The solution to the quadratic programming problem (2.7) gives the vector of control action $\Delta\bar{u}(k)$, which is used to calculate the optimal input signal. Since the objective function in is strictly convex (due to W being positive definite), and the feasible region of linear constraints is a closed convex set, the solution to the quadratic programming problem is unique and satisfies the Karush-Kuhn-Tucker (KKT) optimality conditions [51].

2.1.2 Linear Programming Formulation

Although the quadratic criterion is popular and has been widely used in various MPC applications. Several MPC algorithms using linear programming have been presented. For example, Zadeh and Whalen [52] and Propoi [53] introduce the approaches to solve MPC problem based on linear programming in the early sixties. And some other authors published their investigation concerning the linear programming based MPC [54]-[57].

In this section, we formulate MPC as a standard linear programming problem. With l_1 criterion, MPC can be formulated other than (2.3) as follow:

$$\begin{aligned} &\min \sum_{j=1}^{N} Q[r(k+j|k) - y(k+j|k)] + \sum_{j=0}^{N_u-1} R\Delta u(k+j|k) \\ &\text{s.t.} \quad u_{\min} \le u(k+j|k) \le u_{\max}, \quad j = 0, ..., N_u - 1; \\ &\Delta\, u_{\min} \le \Delta u(k+j|k) \le \Delta u_{\max}, \quad j = 0, ..., N_u - 1; \\ &y_{\min} \le y(k+j|k) \le y_{\max}, \quad j = 1, ..., N. \end{aligned} \tag{2.8}$$

The optimization problem (2.8) can be further formulated as a standard linear programming problem using the standard method [58].

Define the following vectors:

$$\phi(k) = [\phi, \cdots, \phi]^T \in \Re^{Np},$$

$$\varphi(k) = [\varphi, \cdots, \varphi]^T \in \Re^{N_u m},$$

$$s = [\Delta\bar{u}^T(k) \quad \phi^T(k) \quad \varphi^T(k)]^T \in \Re^{2N_u m + Np},$$

that satisfies

$$\begin{aligned} -\phi(k) &\leq \pm Q[\bar{r}(k) - \bar{y}(k)], \\ -\varphi(k) &\leq \pm R\Delta\bar{u}(k), \end{aligned} \tag{2.9}$$

where $\pm$ means that the constraints is duplicated for each sign. The problem (2.8) can be rewritten in as a standard linear programming problem:

$$\begin{aligned} \min \quad & f^T s \\ \text{s.t.} \quad & h_{\min} \leq Fs \leq h_{\max} \end{aligned} \tag{2.10}$$

where the coefficient matrices and vectors are:

$$h_{\min} = [l_{\min} \quad Q(\bar{r}(k) - Sx(k) - Vu(k-1)) \quad O \quad -\infty]^T,$$

$$h_{\max} = [l_{\max} \quad \infty \quad Q(\bar{r}(k) - Sx(k) - Vu(k-1)) \quad O]^T,$$

$$h_{\min}, h_{\min} \in \Re^{(3N_u+2N)m+2Np},$$

$$f = [0, \cdots, 0, 1, \cdots, 1]^T \in \Re^{2N_u m+Np},$$

$$F = \begin{bmatrix} G & O & O \\ QM & I & O \\ R & O & I \\ QM & -I & O \\ R & O & -I \end{bmatrix},$$

$$F \in \Re^{[(3N_u+2N)m+2Np]\times(2N_u m+Np)},$$

where O denotes the zero matrix.

As linear programming problem (2.10) depends on the current state $x(k)$ and past input $u(k-1)$, we will introduce a neural network to solve the problem at each time interval in the next section.

2.2 Neural Network Approaches

In this section, based on the linear and quadratic programming formulations (2.7) and (2.10) in the previous section, we propose two neural network approaches to MPC.

2.2.1 Neural Network Model 1

Liu and Wang [28] developed a one-layer recurrent neural network called the simplified dual neural network for solving quadratic programming problems, which has showed good performance and lower computational complexity. In this part of the section, we apply the neural network for MPC.

Consider (2.7) as a primal problem, then its dual problem is:

$$\begin{aligned} \min \quad & -\frac{1}{2}v^T W v + l_{\min}^T \alpha - l_{\max}^T \beta \\ \text{s.t.} \quad & Wv + c - G^T\alpha + G^T\beta = 0 \end{aligned} \tag{2.11}$$

where $\alpha \in \Re^{2N_u m + 2Np}$, $\beta \in \Re^{2N_u m + 2Np}$ are dual decision variables.

By defining $z = \alpha - \beta$, the Karush-Kauhn-Tucker condition of (2.7) are:

$$\begin{aligned} & Wv + c - G^T z = 0, \\ & Gv = \lambda(Gv - z), \end{aligned} \tag{2.12}$$

where $\lambda(\cdot)$ is a piecewise linear function, defined as:

$$\lambda(\varepsilon_i) = \begin{cases} l_{\min}, & \varepsilon_i < l_{\min}; \\ \varepsilon_i, & l_{\min} \le \varepsilon_i \le l_{\max}; \\ l_{\max}, & \varepsilon_i > l_{\max}. \end{cases} \tag{2.13}$$

Based on (2.12) and (2.13), the dynamic equation of neural network for solving quadratic programming problem (2.7) can be described as:

- state equation

$$\epsilon \frac{dz}{dt} = -Gv + \lambda(Gv - z), \tag{2.14}$$

- output equation

$$v = W^{-1}(G^T z - c), \tag{2.15}$$

where $z \in \Re^{2N_u m + 2Np}$ is the state vector, ϵ is a scaling parameter that control the convergence rate of the neural network.

According to the convergence analysis in [28], we can ensure that the proposed neural network is Lyapunov stable and globally convergent to the optimal solution.

2.2.2 Neural Network Model 2

It has been shown that linear programming problems can also be solved using neural networks. Recently, a one-layer recurrent neural network with a discontinuous hard-limiting activation function for linear programming was developed [41]. In this paper, we present this neural network for solving (2.10).

To apply the neural network model, let us further formulate (10) as:

$$\begin{aligned} \min \quad & f^T s \\ \text{s.t.} \quad & b_{\min} \leq s \leq b_{\max} \end{aligned} \tag{2.16}$$

where

$$b_{\min} = F^T h_{\min} \in \Re^{N_u(m+1)+N},$$

$$b_{\max} = F^T h_{\max} \in \Re^{N_u(m+1)+N}.$$

According to the KKT conditions, s^* is an optimal solution of (2.16), if and only if there exist a $w^* \in \Re^{2N_u m + Np}$ such that (s^*, w^*) satisfies the following optimality conditions:

$$\begin{aligned} & f + w = 0, \\ & \begin{cases} w_i \geq 0, & s_i = b_{\max(i)}; \\ w_i = 0, & b_{\min(i)} \leq s_i \leq b_{\max(i)}; \\ w_i \leq 0, & s_i = b_{\max(i)}. \end{cases} \end{aligned} \tag{2.17}$$

Based on the above conditions, the dynamic equation of the proposed recurrent neural network model is described as follows:

$$\frac{ds}{dt} = -\epsilon\{\sigma g(s) + f\}, \tag{2.18}$$

where ϵ is a positive scaling constant, σ is a nonnegative gain parameter. $g(\cdot)$ is a discontinuous activation function, defined as:

$$g_i(s_i) = \begin{cases} 1, & s_i > b_{\max(i)}; \\ [0,1], & s_i = b_{\max(i)}; \\ 0, & b_{\min(i)} < s_i < b_{\max(i)}; \\ [-1,0], & s_i = b_{\min(i)}; \\ -1, & s_i < b_{\min(i)}, \end{cases} \tag{2.19}$$

where $i = 1, 2, ..., 2N_u m + Np$.

It is proven in [41] that the neural network is globally convergent to the optimal solution.

2.2.3 Control Scheme

The control scheme based on neural networks can be summarized as follows:

1. Let $k = 1$. Set terminal time T, sample time t, predictive horizon N, control horizon N_u, weighting matrices Q and R.

2. Calculate process model matrices S, V, M, neural network parameters W, G, f, $l_{\max}$, $l_{\min}$, $b_{\max}$, $b_{\min}$.

3. Solve the quadratic and linear programming problems (7) and (10) using the proposed two neural networks, obtaining the optimal control action $\Delta\bar{u}(k)$.

4. Calculate the optimal input vector $u(k) = \Delta u(k|k) + u(k-1)$.

5. If $k < T$, set $k = k + 1$, return to step 2; otherwise, end.

2.3 Simulation Results

Consider a quadruple-tank process described in [27], the objective is to control the level of the two lower tanks y_1 and y_2, using the two pumps u_1 and u_2. The process model of the quadruple-tank system is

$$\begin{aligned}
\dot{x}_1 &= -\frac{a_1}{A_1}\sqrt{2gx_1} + \frac{a_3}{A_1}\sqrt{2gx_3} + \frac{\gamma_1\rho_1}{A_1}u_1, \\
\dot{x}_2 &= -\frac{a_2}{A_2}\sqrt{2gx_2} + \frac{a_4}{A_2}\sqrt{2gx_4} + \frac{\gamma_2\rho_2}{A_2}u_2, \\
\dot{x}_3 &= -\frac{a_3}{A_3}\sqrt{2gx_3} + \frac{(1-\gamma_2)\rho_2}{A_3}u_2, \\
\dot{x}_4 &= -\frac{a_4}{A_4}\sqrt{2gx_4} + \frac{(1-\gamma_1)\rho_1}{A_1}u_1, \\
y_1 &= \rho_c x_1, \quad y_2 = \rho_c x_2,
\end{aligned} \tag{2.20}$$

where x_i is the water level in tank i. Choose the controller parameters as follows: cross-section of tank $A_1 = A_3 = 28, A_2 = A_4 = 32$; cross-section of the outlet hole $a_1 = a_3 = 0.071, a_2 = a_4 = 0.057$; acceleration due to gravity $g = 981$; gains $\rho_c = 0.5$; the fraction of water flowing to tank from pump $\gamma_1 = 0.7, \gamma_2 = 0.6$; prediction horizon $N = 10$; control horizon $N_u = 2$; sampling time $t = 1[s]$; weighting matrices $Q = I, \quad R = 10I$.

The process model is linearized around the operational points $x_{1o} = 12.4$, $x_{2o} = 12.7$, $x_{3o} = 1.8$, $x_{4o} = 1.4$. Consider input constraints $u_{\min} = 0$, $u_{\max} = 6$; output constraints $y_{\min} = 0$, $y_{\max} = 7.5$.

The simulation results are showed in Figs. 2.1 - 2.4. At time $k = 20$, a step reference change is commanded in y_1. At the same time a step load disturbance enters in y_2. We can see that both the proposed neural network approaches are effective, based on the advantages in parallel computation and hardware implementation of neural network, the proposed approaches can solve the problem in real time efficiently.

The neural network controller based on linear programming (NN2) responds faster than the one based on quadratic programming (NN1). That is because NN2 approach have less computational cost than NN1 approach. We can conclude that

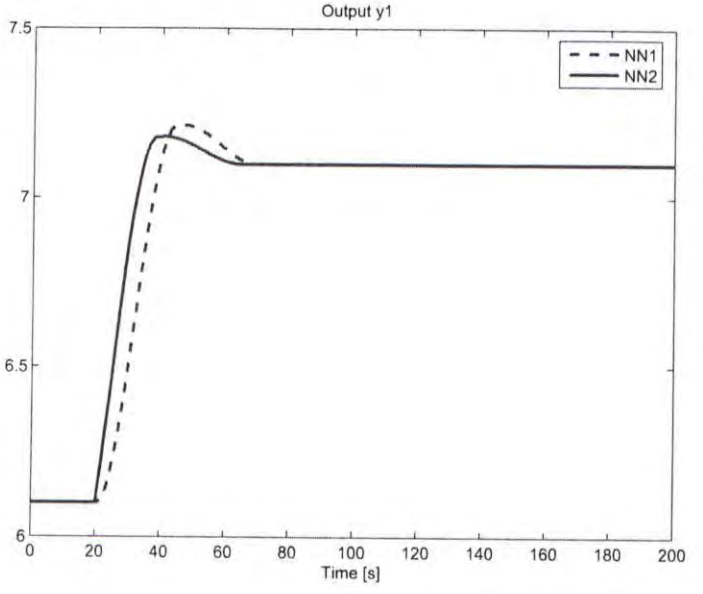

Figure 2.1: Output responses in tank 1 of NN1 and NN2 approaches.

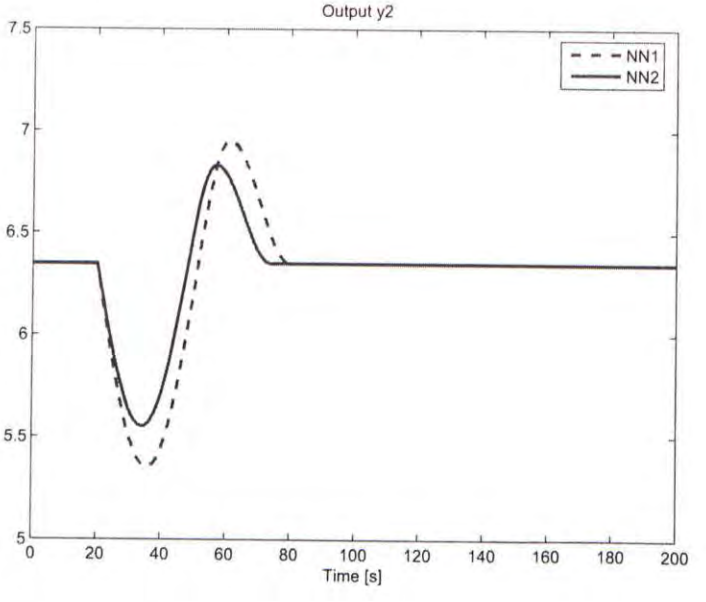

Figure 2.2: Output responses in tank 2 of NN1 and NN2 approaches

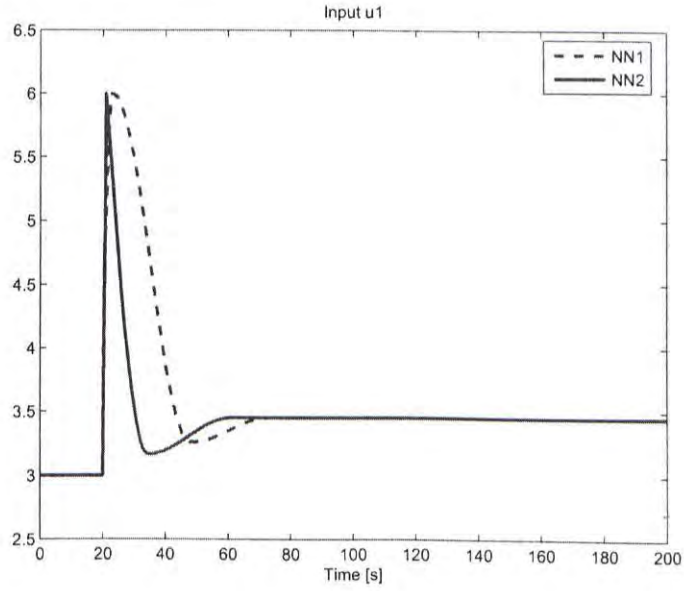

Figure 2.3: Input responses in tank 1 of NN1 and NN2 approaches.

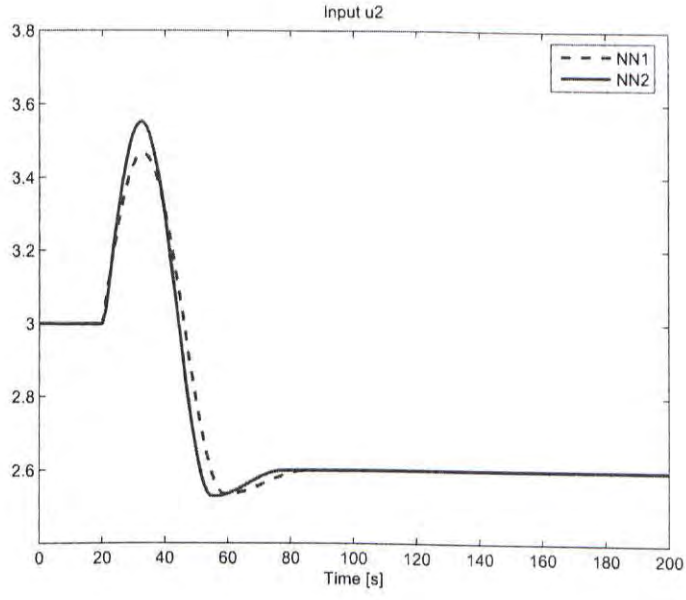

Figure 2.4: Input responses in tank 2 of NN1 and NN2 approaches.

the NN2 approach is more suitable for solving control problems with large size and stringent real-time requirement. However, using approaches based on linear programming may result in a poor control performance, which depends on the the selection of the weighting matrices Q and R [59].

□ **End of chapter.**

Chapter 3

Model Predictive Control for Nonlinear Affine Systems Based on the Simplified Dual Neural Network

3.1 Problem Formulation

Consider a discrete-time nonlinear affine system:

$$\begin{aligned} x(k+1) &= f(x(k)) + g(x(k))u(k), \\ y(k) &= Ax(k), \end{aligned} \tag{3.1}$$

where $x(k) \in \Re^n$ is the state vector; $u(k) \in \Re^m$ is the input vector; $y(k) \in \Re^p$ is the output vector; $f(\cdot)$ and $g(\cdot)$ are nonlinear functions; $A \in \Re^{p \times n}$. The system is subject to the following constraints:

$$u_{\min} \leq u(k) \leq u_{\max}, \quad |\Delta u(k)| \leq \Delta u_{\max}, \quad y_{\min} \leq y(k) \leq y_{\max}, \tag{3.2}$$

where $u_{\min} \leq u_{\max}$, $y_{\min} \leq y_{\max}$, and $\Delta u_{\max} > 0$ are vectors of upper and lower bounds; $\Delta u(k+j|k) = u(k+j|k) - u(k+j-1|k)$ denotes the input increment.

For the above process model, the following cost function can be formulated and used for calculation of the optimal input trajectory over the control horizon:

$$J(k) = \sum_{j=1}^{N} \|r(k+j|k) - y(k+j|k)\|_Q^2 + \sum_{j=0}^{N_u-1} \|\Delta u(k+j|k)\|_R^2, \tag{3.3}$$

where $r(k+j|k)$ denotes the reference trajectory of output signal (desired output) at sampling instant k, $y(k+j|k)$ denotes the predicted output, N is the predictive horizon ($1 \leq N$), N_u denotes the control horizon ($0 < N_u \leq N$), $Q \in \Re^{p\times p}$ and $R \in \Re^{m\times m}$ are appropriate weighting matrices, and $\|\cdot\|_Q$ and $\|\cdot\|_R$ are weighted norms defined as $\|z\|_W = \sqrt{z^T W z}$.

In order to obtain the formulation of the predicted state vectors, the previous predicted state vectors are used, which can be computed as follows:

$$\begin{aligned}
x(k+1|k) &= f(x(k)) + g(x(k))(u(k-1) + \Delta u(k|k)),\\
x(k+2|k) &= f(x(k+1|k-1)) + g(x(k+1|k-1))(u(k-1)\\
&\quad + \Delta u(k|k) + \Delta u(k+1|k)),\\
&\vdots\\
x(k+N|k) &= f(x(k+N-1|k-1)) + g(x(k+N-1|k-1))\\
&\quad (u(k-1) + \Delta u(k|k) \cdots + \Delta u(k+N-1|k)).
\end{aligned} \tag{3.4}$$

Define following vectors:

$$\begin{aligned}
\bar{y}(k) &= [y(k+1|k) \quad \cdots \quad y(k+N|k)]^T \in \Re^{Np},\\
\bar{x}(k) &= [x(k+1|k) \quad \cdots \quad x(k+N|k)]^T \in \Re^{Nn},\\
\Delta\bar{u}(k) &= [\Delta u(k|k) \quad \cdots \quad \Delta u(k+N_u-1|k)]^T \in \Re^{N_u m}.
\end{aligned}$$

The predicted output depends on the previous predicted states, past input, and assumed value of current state. It can be expressed in a more concise form:

$$\bar{y}(k) = \tilde{A}\bar{x}(k) = \tilde{A}(G(k-1)\Delta\bar{u}(k) + K(k-1) + F(k-1)) \tag{3.5}$$

where

$$\tilde{A} = \begin{bmatrix} A & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A \end{bmatrix} \in \Re^{Np\times Nn},$$

$$G(k-1) = \begin{bmatrix} g(x(k-1)) & \cdots & 0 \\ g(x(k+1|k-1)) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ g(x(k+N-1|k-1)) & g(x(k+N-1|k-1)) & g(x(k+N-1|k-1)) \end{bmatrix} \in \Re^{Nn \times Nm},$$

$$K(k-1) = \begin{bmatrix} g(x(k))u(k-1) \\ g(x(k+1|k-1))u(k-1) \\ \vdots \\ g(x(k+N-1|k-1))u(k-1) \end{bmatrix} \in \Re^{Nn},$$

$$F(k-1) = \begin{bmatrix} f(x(k)) \\ f(x(k+1|k-1)) \\ \vdots \\ f(x(k+N-1|k-1)) \end{bmatrix} \in \Re^{Nn}.$$

Thus, the optimization problem associated with MPC can be expressed as:

$$\begin{aligned} \min \quad & \|\bar{r}(k) - \tilde{A}F(k-1) - \tilde{A}K(k-1) - \tilde{A}G(k-1)\Delta\bar{u}(k)\|_Q^2 \\ & + \|\Delta\bar{u}(k)\|_R^2, \\ \text{s.t.} \quad & -\Delta\bar{u}_{\max} \leq \Delta\bar{u}(k) \leq \Delta\bar{u}_{\max}, \\ & \bar{u}_{\min} \leq \bar{u}(k-1) + H\Delta\bar{u}(k) \leq \bar{u}_{\max}, \\ & \bar{y}_{\min} \leq \tilde{A}(F(k-1) + K(k-1) + G(k-1)\Delta\bar{u}(k)) \leq \bar{y}_{\max}, \end{aligned} \tag{3.6}$$

where

$$\begin{aligned} \bar{u}(k) &= [u(k) \quad \cdots \quad u(k)] \in \Re^{N_u m}, \\ \bar{r}(k) &= [r(k+1|k) \quad \cdots \quad r(k+N|k)]^T \in \Re^{Np}, \\ \Delta\bar{u}_{\max} &= [\Delta u_{\max} \quad \cdots \quad \Delta u_{\max}]^T \in \Re^{N_u m}, \\ \bar{u}_{\min} &= [u_{\min} \quad \cdots \quad u_{\min}]^T \in \Re^{N_u m}, \\ \bar{u}_{\max} &= [u_{\max} \quad \cdots \quad u_{\max}]^T \in \Re^{N_u m}, \\ \bar{y}_{\min} &= [y_{\min} \quad \cdots \quad y_{\min}]^T \in \Re^{Np}, \\ \bar{y}_{\max} &= [y_{\max} \quad \cdots \quad y_{\max}]^T \in \Re^{Np}, \end{aligned}$$

$$H = \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & I & I \end{bmatrix} \in \Re^{N_u m \times N_u m}.$$

By defining the variable vector $v = \Delta\bar{u}(k) \in \Re^{N_u m}$, the optimization problem (3.6) can be rewritten as a QP problem:

$$\begin{aligned} \min \quad & \frac{1}{2} v^T W v + c^T v, \\ \text{s.t.} \quad & l_{\min} \leq Ev \leq l_{\max}, \end{aligned} \tag{3.7}$$

where the coefficient matrices and vectors are

$$l_{\min} = [-\infty \quad \Delta\bar{u}_{\min}]^T \in \Re^{3N_u m + 2Np},$$

$$l_{\max} = [b \quad \Delta\bar{u}_{\max}]^T \in \Re^{3N_u m + 2Np},$$

$$W = 2(\tilde{A}G(k-1)^T Q \tilde{A}G(k-1) + R) \in \Re^{N_u m \times N_u m},$$

$$c = -2\tilde{A}G(k-1)^T Q(\bar{r}(k) - K(k-1) - F(k-1)) \in \Re^{N_u m},$$

$$E = [-H \quad H \quad -\tilde{A}G(k-1) \quad \tilde{A}G(k-1) \quad I]^T \in \Re^{(3N_u m + 2Np) \times N_u m},$$

$$b = \begin{bmatrix} -\bar{u}_{\min} + \bar{u}(k-1) \\ \bar{u}_{\max} - \bar{u}(k-1) \\ -\bar{y}_{\min} + \tilde{A}F(k-1) + \tilde{A}K(k-1) \\ \bar{y}_{\max} - \tilde{A}F(k-1) - \tilde{A}k(k-1) \end{bmatrix} \in \Re^{2N_u m + 2Np}.$$

The solution to the QP problem (3.7) gives the vector of incremental control action $\Delta\bar{u}(k)$, which is used to calculate the optimal control input. Since the objective function is strictly convex (due to W being positive definite), and the feasible region of linear constraints is a convex set, the solution to the QP problem is unique and satisfies the Karush-Kuhn-Tucker (KKT) optimality conditions.

3.2 A Neural Network Approach

Based on the QP formulation (3.7) in the previous section, we employ the simplified dual network [28] for controller design.

3.2.1 The Simplified Dual Network

Consider (3.7) as a primal problem, then its dual problem is:

$$\begin{aligned} \min \quad & -\frac{1}{2}v^T W v + l_{\min}^T \alpha - l_{\max}^T \beta \\ \text{s.t.} \quad & Wv + c - E^T\alpha + E^T\beta = 0 \end{aligned} \tag{3.8}$$

where $\alpha \in \Re^{3N_u m+2Np}$, $\beta \in \Re^{3N_u m+2Np}$ are dual decision vectors.

By defining $z = \alpha - \beta$, the Karush-Kauhn-Tucker condition of (3.7) are:

$$\begin{aligned} & Wv + c - E^T z = 0, \\ & Ev = P(Ev - z), \end{aligned} \tag{3.9}$$

where $P(\cdot)$ is a piecewise linear function, defined as:

$$P(\varepsilon) = \begin{cases} l_{\min}, & \varepsilon < l_{\min}; \\ \varepsilon, & l_{\min} \le \varepsilon \le l_{\max}; \\ l_{\max}, & \varepsilon > l_{\max}. \end{cases} \tag{3.10}$$

Based on (3.9) and (3.10), the dynamic equation of the simplified dual network for solving QP problem (3.7) can be described as:

- state equation

$$\frac{dz}{dt} = \lambda(-Gv + P(Ev - z)), \tag{3.11}$$

- output equation

$$v = W^{-1}(E^T z - c), \tag{3.12}$$

where $z \in \Re^{3N_u m+2Np}$ is the state vector, λ is a scaling parameter that control the convergence rate of the neural network. The simplified dual network is composed of one layer of $3N_n m+2Np$ neurons. Compared with other models, this RNN has a simpler structure. According to the convergence analysis in [28], we can ensure that the simplified dual network is Lyapunov stable and globally convergent to the optimal solution at each time interval.

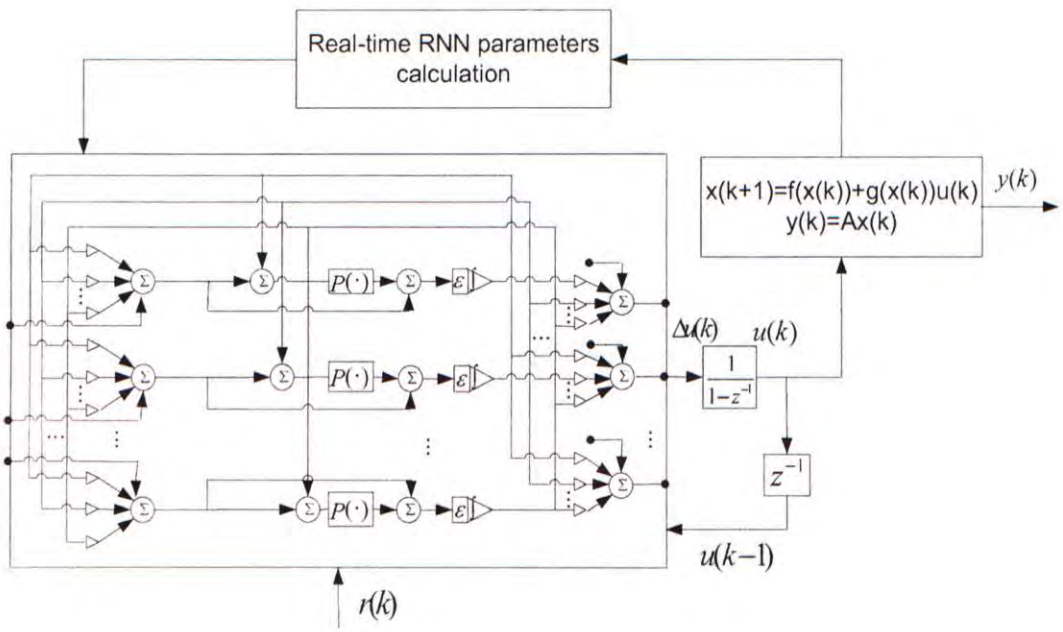

Figure 3.1: Block diagram of the RNN-based MPC scheme

3.2.2 RNN-based MPC Scheme

The MPC scheme for nonlinear affine systems based on the simplified dual network (a block diagram is shown in Fig. 3.1, with z^{-1} being the one-step backward shift operator) can be summarized as follows:

1. Let $k = 1$. Set terminal time T, sample time t, predictive horizon N, control horizon N_u, weighting matrices Q and R.
2. Calculate process model matrices and neural network parameters E, W, G, f, $l_{\max}$, $l_{\min}$.
3. Solve the QP problem (5.7) using the proposed RNN, obtaining the optimal control action $\Delta\bar{u}(k)$.
4. Calculate the optimal control vector $u(k) = \Delta u(k|k) + u(k-1)$.
5. Set $k = k + 1$, return to step 1.

The proposed RNN-based MPC scheme operates in a massively parallel fashion, which is suitable for large-scale system implementation.

3.3 Simulation Results

In this section, simulation results in three numerical examples are provided to illustrate the performance of the proposed RNN-based MPC scheme.

3.3.1 Example 1

Consider a single-input single-output nonlinear system:

$$\begin{aligned} y(k+1) = & \frac{1.5y(k)y(k-1)}{1+y^2(k)+y^2(k-1)} + 0.35\sin[y(k)+y(k-1)] \\ & + \{1.2+\cos[y(k)+y(k-1)]\}u(k), \end{aligned} \tag{3.13}$$

subject to

$$0 \leq y \leq 2.1, \quad 0.5 \leq u \leq 1.8. \tag{3.14}$$

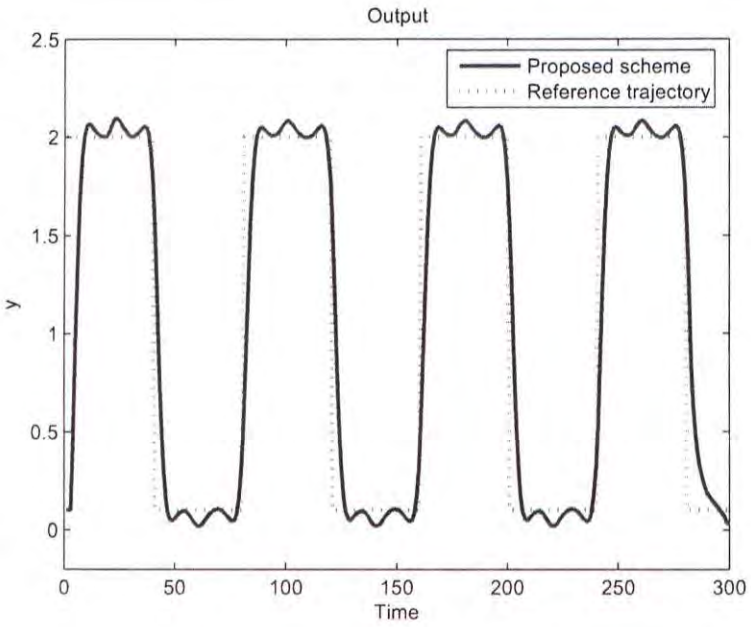

Figure 3.2: Output in Example 1

The objective is to control the system to track a reference trajectory (a square wave). The RNN-based controller parameters are: prediction horizon $N = 15$, control horizon $N_u = 10$, weighting matrices $Q = R = 5I$, sampling frequency is 10Hz. Simulation results are shown in Figs. 3.2-3.3. From the output response in Fig. 3.2, we can see that the proposed approach gives a good tracking performance with stable and fast responses with no variable exceeding the given constraints.

3.3.2 Example 2

Ball and plate is commonly created for control system modeling, design, implementation, and verification as a training tool for science and engineering practice.

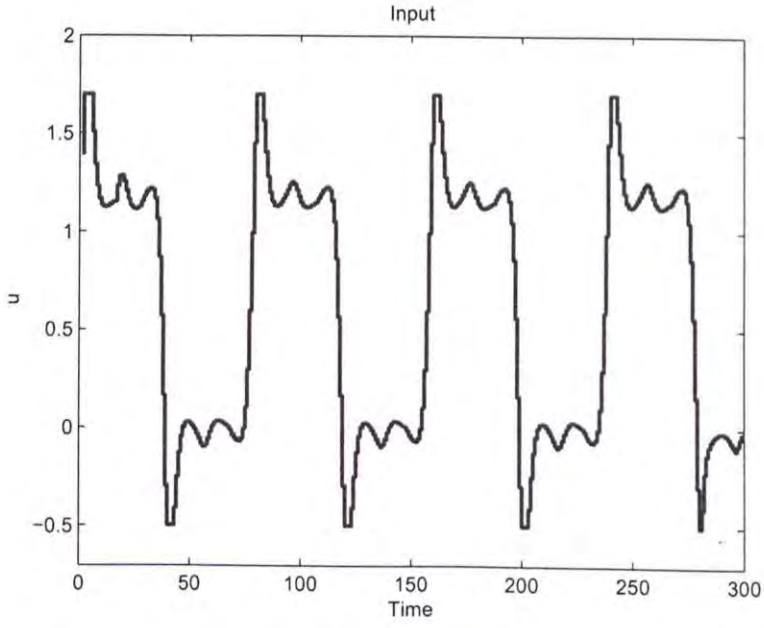

Figure 3.3: Input in Example 1

From the literature, the typical devices for the ball and plate system include a rigid plate with a ball rolling freely on the plate, several linear sensors for locating ball position and detecting plate deflection angle, torque generation facilities such as stepping motor, servo motor, gearbox, belt, etc.

Consider a ball and plate system, which can be described as the following

MIMO nonlinear affine system:

$$
\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \\ \dot{x_5} \\ \dot{x_6} \\ \dot{x_7} \\ \dot{x_8} \end{bmatrix} = \begin{bmatrix} x_2 \\ 0.714(x_1 x_4^2 + x_4 x_5 x_8 - 9.81 \sin x_3) \\ x_4 \\ 0 \\ x_6 \\ 0.714(x_5 x_8^2 + x_1 x_4 x_8 - 9.81 \sin x_7) \\ x_8 \\ 0 \end{bmatrix}
$$

$$
+ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} u_x \\ u_y \end{bmatrix},
$$

$$
y_x = x_1, \quad y_y = x_5,
$$

where x_1 and x_5 are the positions of the ball on X-axis and Y-axis, x_2 and x_6 are ball speeds on X-axis and Y-axis, x_3 and x_7 are the plate deflection angles, x_4 and x_8 are the angular velocities, control u_x and u_y are the angular accelerations. The model is discretized using the Euler method.

The objective is to place the ball at the desired position on the plate. The initial state is $[0, 0, 0, 0, 0, 0, 0, 0]$, desired position is $[0.1, 0.2]$. The parameters of the RNN-based controller are chosen as same as those in Example 3.1. As shown in Fig. 3.4, the proposed RNN-based controller is capable of moving the ball to the desired position.

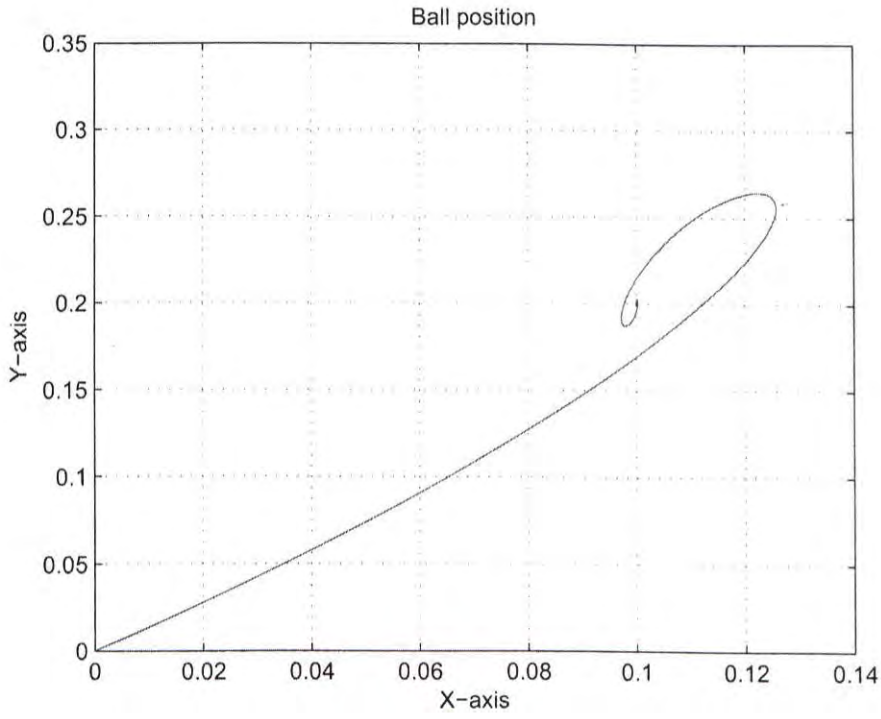

Figure 3.4: Outputs of the ball and plate system in Example 2

3.3.3 Example 3

Consider a continuous stirred tank reactor (CSTR), which is a single-input multi-output nonlinear model in process industries [77]:

$$g(x(k)) = \begin{bmatrix} 0 \\ \beta \end{bmatrix}, \qquad f(x(k)) =$$

$$\begin{bmatrix} x_1(k) + T_s[-\alpha x_1(k) + D_a(1 - x_1(k))e^{\frac{x_2(k)}{1-x_2(k)/\gamma}}] \\ x_2(k) + T_s[-\alpha x_1(k) + D_a(1 - x_1(k))e^{\frac{x_2(k)}{1-x_2(k)/\gamma}} - \beta x_2(k)] \end{bmatrix},$$

The RNN-based controller parameters are: $A = I$, $\alpha = 1, \beta = 0.3, \gamma = 20, B = 1, D = 0.072, T = 0.1$. The system is subject to the following constraints:

$$0 \leq y_1 \leq 0.31, \quad 0 \leq y_2 \leq 2, \quad 7 \leq u \leq 18.$$

In this simulation, the initial outputs are [0, 0], set points to be tracked (desired outputs) are [0.30, 1.97], the prediction and control horizons are $N = 10$ and $N_u = 5$, weighting matrices are $Q = I$, $R = 5I$, sampling frequency is 10Hz.

To demonstrate the superiority of the proposed RNN-based approach, both linear MPC (based on linearization) and nonlinear MPC scheme in [50] were applied to the process model. Simulation results are shown in Figs. 3.5-3.7. As shown in Figs. 3.5 and 3.6, the proposed approach gives a better performance than the other two approaches with fastest set-point tracking rate and no variable exceeding the constraints.

□ **End of chapter.**

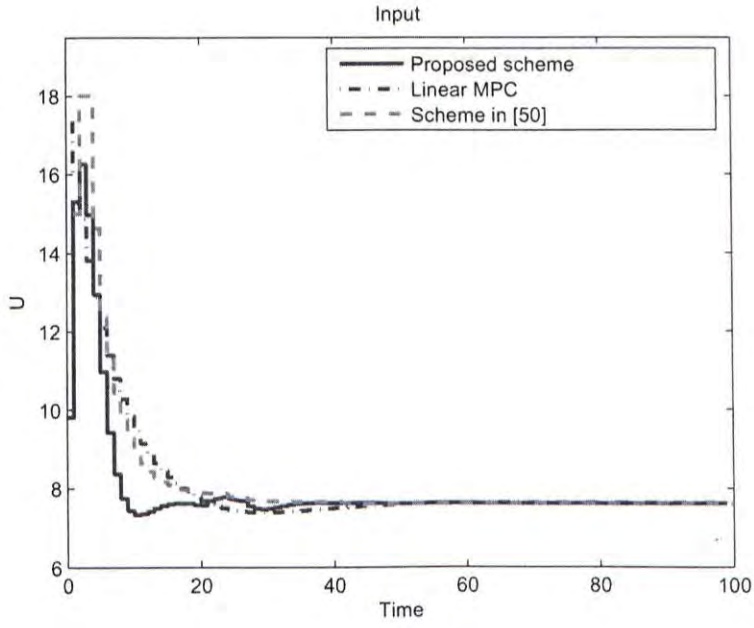

Figure 3.5: Output 1 in Example 3

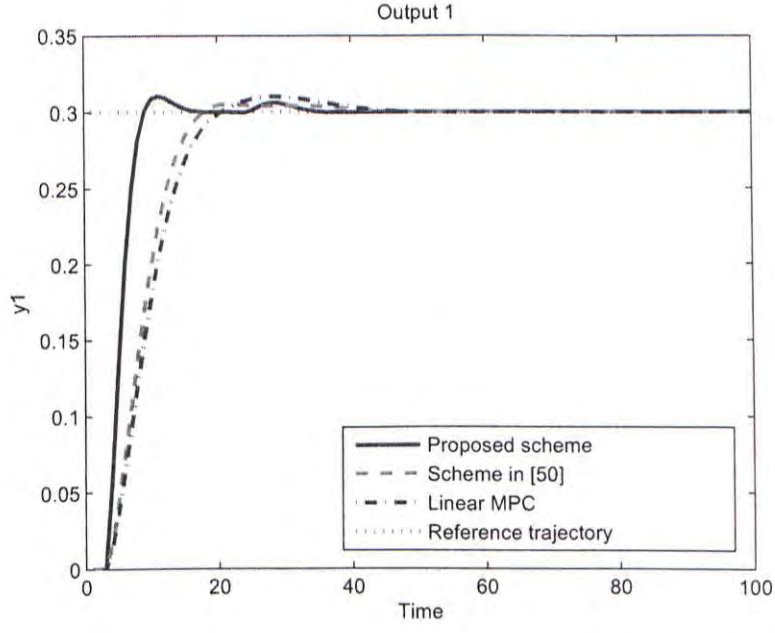

Figure 3.6: Output 2 in Example 3

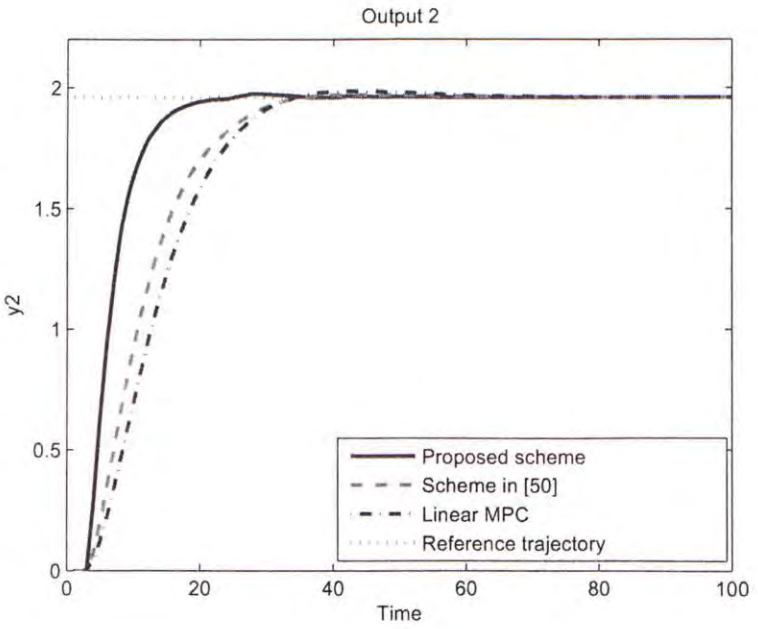

Figure 3.7: Input in Example 3

Chapter 4

Nonlinear Model Predictive Control Using a Recurrent Neural Network

4.1 Problem Formulation

Consider the following nonlinear system:

$$y(k+1) = f(y(k), u(k)), \tag{4.1}$$

where $f(\cdot)$ is a continuously differentiable nonlinear function; $y(k) \in \Re^n$ and $u(k) \in \Re^m$ are output and input vectors, respectively. By Taylor series, the above nonlinear system can be decomposed around reference point $[x_r(k), y_r(k)]$ into a linear system plus a unknown nonlinear term:

$$\begin{aligned} y(k+1) =& f(y_r(k), u_r(k)) + \frac{\partial f(y,u)}{\partial y} |_{y_r(k),u_r(k)} (y(k) - y_r(k)) + \\ & \frac{\partial f(y,u)}{\partial u} |_{y_r(k),u_r(k)} (u(k) - u_r(k)) + \varepsilon(y(k), u(k)), \end{aligned} \tag{4.2}$$

where $\varepsilon(y(k), u(k))$ is the unknown high-order term of the Taylor series. Since the reference trajectory is:

$$y_r(k+1) = f(y_r(k), u_r(k)) \tag{4.3}$$

Then subtract (4.3) from (4.2)

$$\begin{aligned} y(k+1) - y_r(k+1) = & \frac{\partial f(y,u)}{\partial y} \mid_{y_r(k),u_r(k)} (y(k) - y_r(k)) + \\ & \frac{\partial f(y,u)}{\partial u} \mid_{y_r(k),u_r(k)} (u(k) - u_r(k)) + \varepsilon(y(k), u(k)) \end{aligned} \tag{4.4}$$

By defining $y'(k) = y(k) - y_r(k)$ and $u'(k) = u(k) - u_r(k)$, (4.4) becomes

$$y'(k+1) = \frac{\partial f(y,u)}{\partial y} \mid_{y_r(k),u_r(k)} y'(k) + \frac{\partial f(y,u)}{\partial u} \mid_{y_r(k),u_r(k)} u'(k) + \varepsilon(y(k), u(k)), \tag{4.5}$$

Denote $\varepsilon(k) = \varepsilon(y(k), u(k))$ for brevity. The above equation can be rewritten as:

$$y'(k+1) = Ay'(k) + Bu'(k) + \varepsilon(k) \tag{4.6}$$

where

$$A = \frac{\partial f(y,u)}{\partial y} \mid_{y_r(k),u_r(k)}, \quad B = \frac{\partial f(y,u)}{\partial u} \mid_{y_r(k),u_r(k)},$$

The optimization problem associated with nonlinear MPC is

$$\begin{aligned} \min \quad & \sum_{j=1}^{N} [y'(k+j|k)]^T Q [y'(k+j|k)] + \sum_{j=0}^{N_u-1} \Delta u(k+j|k)^T R \Delta u(k+j|k) \\ \text{s.t.} \quad & u_{\min} \le u(k+j|k) \le u_{\max}, \quad j = 0, ..., N_u - 1; \\ & \Delta\, u_{\min} \le \Delta u(k+j|k) \le \Delta u_{\max}, \quad j = 0, ..., N_u - 1; \\ & y_{\min} \le y'(k+j|k) \le y_{\max}, \quad j = 1, ..., N; \end{aligned} \tag{4.7}$$

where k is the current time step. $\Delta u(k+j|k)$ denote the input increment, where $\Delta u(k+j|k) = u(k+j|k) - u(k-1+j|k)$. N is the predictive horizon, where $1 \le N$. N_u denote the control horizon, where $0 < N_u \le N$. $Q \in \Re^{n \times n}$ and $R \in \Re^{m \times m}$ are appropriate weighting matrices.

Define following vectors:

$$y(k) = [y(k+1|k) \quad \cdots \quad y(k+N|k)]^T \in \Re^{Nn},$$

$$\bar{u}(k) = [u(k|k) \quad \cdots \quad u(k+N_u-1|k)]^T \in \Re^{N_u m},$$

$$\Delta\bar{u}(k) = [\Delta u(k|k) \quad \cdots \quad \Delta u(k+N_u-1|k)]^T \in \Re^{N_u m},$$

$$\bar{\varepsilon}(k) = [\varepsilon(k+1|k) \quad \cdots \quad \varepsilon(k+N|k)]^T \in \Re^{Nm},$$

The predicted output $\bar{y}(k)$ are expressed in the following form:

$$\begin{aligned} \bar{y}(k) &= S(k)y(k) + M(k)\bar{u}(k) + \bar{\varepsilon}(k) \\ &= S(k)y(k) + M(k)\Delta\bar{u}(k) + V(k)u(k-1) + \bar{\varepsilon}(k), \end{aligned} \tag{4.8}$$

where

$$S(k) = [A(k) \quad A(k)^2 \quad \cdots \quad A(k)^N]^T \in \Re^{Nn\times n},$$

$$V(k) = \begin{bmatrix} B(k) \\ (A(k)+I)B(k) \\ \vdots \\ (A(k)^{N_u-1} + \cdots + A(k) + I)B(k) \\ (A(k)^{N_u} + \cdots + A(k) + I)B(k) \\ \vdots \\ (A(k)^{N-1} + \cdots + A(k) + I)B(k) \end{bmatrix} \in \Re^{Nm\times m},$$

$$M(k) = \begin{bmatrix} B(k) & \dots & 0 \\ (A(k)+I)B(k) & \dots & 0 \\ \vdots & \ddots & \vdots \\ (A(k)^{N_u-1} + ...I)B(k) & \dots & B(k) \\ (A(k)^{N_u} + ...I)B(k) & \dots & (A(k)+I)B(k) \\ \vdots & \ddots & \vdots \\ (A(k)^{N-1} + ...I)B(k) & \dots & (A(k)^{N-N_u} + ...I)B(k) \end{bmatrix} \in \Re^{Nm\times N_u n},$$

where I denotes the identity matrix. Define:

$$\Delta\bar{u}_{\max} = [\Delta u_{\max} \cdots \Delta u_{\max}]^T \in \Re^{N_u m},$$

$$\bar{u}_{\min} = [u_{\min} \cdots u_{\min}]^T \in \Re^{N_u m}, \quad \bar{u}_{\max} = [u_{\min} \cdots u_{\max}]^T \in \Re^{N_u m},$$

$$\bar{y}_{\min} = [y_{\min} \cdots y_{\min}]^T \in \Re^{Nn}, \quad \bar{y}_{\max} = [y_{\min} \cdots y_{\max}]^T \in \Re^{Nn},$$

$$H = \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & I & I \end{bmatrix} \in \Re^{N_u m \times N_u m}.$$

Thus, the original optimization problem can be expressed in the following form:

$$\begin{aligned} \min \quad & [S(k)x(k) - M(k)\Delta\bar{u}(k) - V(k)u(k-1) - \bar{\varepsilon}(k)]^T Q[S(k)x(k) - \\ & M(k)\Delta\bar{u}(k) - V(k)u(k-1) - \bar{\varepsilon}(k)] + \Delta\bar{u}^T(k)R\Delta\bar{u}(k) \\ \text{s.t.} \quad & \bar{u}_{\min} \le \bar{u}(k) + H\Delta\bar{u}(k) \le \bar{u}_{\max} \\ & \Delta\bar{u}_{\min} \le \Delta\bar{u}(k) \le \Delta\bar{u}_{\max} \\ & \bar{y}_{\min} \le \bar{y}(k) + M(k)\Delta\bar{u}(k) \le \bar{y}_{\max} \end{aligned} \tag{4.9}$$

By defining the variable vector $v = \Delta\bar{u}(k) \in \Re^{N_u m}$, the original optimization problem can be rewritten as a standard quadratic programming problem form:

$$\begin{aligned} \min \quad & \frac{1}{2}v^T W v + c^T v \\ \text{s.t.} \quad & l_{\min} \le Ev \le l_{\max} \end{aligned} \tag{4.10}$$

where the coefficient matrices and vectors are

$$W = 2(M^T Q M + R) \in \Re^{N_u m \times N_u m},$$

$$c = -2M^T Q(Sx(k) - Vu(k-1) - \bar{\varepsilon}(k)) \in \Re^{N_u m},$$

$$E = [-H \quad H \quad -M \quad M]^T \in \Re^{(2N_u m + 2Nn) \times N_u m},$$

$$l_{\min} = \begin{bmatrix} \bar{u}_{\min} - \bar{u}(k-1) \\ \bar{x}_{\min} - S(k)x(k) - M(k)\Delta\bar{u}(k) - V(k)u(k-1) - \bar{\varepsilon}(k) \end{bmatrix}$$

$$l_{\max} = \begin{bmatrix} \bar{u}_{\max} - \bar{u}(k-1) \\ \bar{x}_{\max} - S(k)x(k) - M(k)\Delta\bar{u}(k) - V(k)u(k-1) - \bar{\varepsilon}(k) \end{bmatrix}$$

If Q and R are positive definite, then W is positive definite, which means the objective function of (4.10) is strictly convex (due to W being positive definite), and the feasible region of linear constraints is a convex set, so the solution to the QP problem (4.10) is unique and satisfies the Karush-Kahn-Tucker (KKT) optimality conditions.

The solution to (4.10) gives the vector of control action $\Delta\bar{u}(k)$, which is used to calculate the optimal input vector. It should be noticed that the vector $\bar{\varepsilon}(k)$ is unknown, thus, the vectors c, $l_{\min}$, and $l_{\max}$ are unknown. In the next section, we propose an RNN approach to solving this problem.

4.2 A Recurrent Neural Network Approach

4.2.1 Neural Network Model

The simplified dual network [28] is applied in predictive controller design. The dynamic equation of the neural network is:

- State equation

$$\frac{dw}{dt} = \lambda(-EW^{-1}E^T w + g(EW^{-1}E^T w - EW^{-1}c - w) + EW^{-1}c), \tag{4.11}$$

- Output equation

$$v = W^{-1}E^T w - W^{-1}c, \tag{4.12}$$

where $\lambda > 0$ is a scaling constant, $w \in \Re^{2N_u m + 2Nn}$ is the state vector. $g(\cdot)$ is a piecewise linear activation function, defined as:

$$g(\mu) = \begin{cases} l_{\min}, & \mu < l_{\min}; \\ \mu, & l_{\min} \leq \mu \leq l_{\max}; \\ l_{\max}, & \mu > l_{\max}. \end{cases} \tag{4.13}$$

The above RNN model can be implemented with a single-layer structure of $2N_u m + 2Nn$ neurons. According to the convergence analysis in [28], the simplified

dual network is Lyapunov stable and globally convergent to the optimal solution to (4.10). However, the nonlinear term of Taylor series $\bar{\varepsilon}(k)$ is still unknown. Thus, the RNN parameters c, $l_{\min}$, and $l_{\max}$ that contain $\bar{\varepsilon}(k)$ are unknown. As a result, the RNN (4.11) and (4.12) can not be applied in MPC design directly and a learning algorithm for estimating unknown parameters is necessary.

4.2.2 Learning Algorithm

Similar to (5.3.2), a recursive learning algorithm can be applied to estimate the unknown RNN parameters.

1. Initialization: Set $\bar{\varepsilon}(0) = 0$, calculate $c(0)$, $l_{\min}(0)$, $l_{\max}(0)$, W, and E.

2. Calculate $\bar{\varepsilon}(i+1))$ as:

$$\bar{\varepsilon}(i+1) = \bar{\varepsilon}(i) + \delta(i)(\hat{x}_k(i) - \bar{x}_k(i)), \tag{4.14}$$

   where $\delta(i)$ is a positive and decreasing convergence factor $\delta(i) < 1$ (choose $\delta(i) = 1/i$ in this example), $\hat{x}_k$ is the state of the original mobile robot kinematic model.

3. Update RNN parameters $c(i+1)$, $l_{\min}(i+1)$, and $l_{\max}(i+1)$ with $\bar{\varepsilon}(i+1)$.

4. Solve the QP problem (4.6) using the proposed neural network model (4.11) and (4.12) to obtain the control action $\Delta\bar{u}_k(i+1)$.

5. If $|\bar{\varepsilon}(i+1) - \bar{\varepsilon}(i)| < \gamma$ for some small positive γ, stop; otherwise, $i = i+1$, return to Step 2 for another iteration.

The convergence property of this algorithm will be proved in 5.3.3.

4.2.3 Control Scheme

The RNN-based nonlinear MPC scheme can be summarized as follow:

1. Let $k = 1$. Set terminal time T, sample time t, predictive horizon N, control horizon N_u, weighting matrices Q and R.

2. Calculate process model matrices and neural network parameters (initial).

3. Obtain a control action $\Delta\bar{u}(k)$ using the simplified dual network and the proposed learning algorithm 4.2.2.

4. Calculate and apply the control input $u(k) = \Delta u(k|k) + u(k-1)$.

5. Set $k = k + 1$, return to step 3.

4.3 Application to Mobile Robot Tracking

In the past decade, mobile robots have been a active area of research and development. Numerous practical applications can be uniquely addressed by mobile robots due to their ability to work in large domains. Specifically, mobile robot have been employed for applications such as room cleaning, disabled people assistance, and factory automation. These applications require mobile robots to have the ability to track the path stably. Based on the wide range of applications, mobile robot research is multidisciplinary by nature, which requires accurate sensing of the environment, intelligent trajectory planning and high precision control. In this chapter, the object is to develop high precision control scheme for mobile robot tracking. In recent years, mobile robot tracking control has been widely investigated by many researchers [78]-[83].

However, in realistic implementations it is difficult to obtain good performance, due to the nonlinearity of the kinematic model and constraints on both input and output. In this chapter, the RNN-based MPC scheme is applied for mobile robot controller design. A two-wheeled mobile robot is considered here.

A mobile robot made up of a rigid body and non deforming wheels is considered (see Fig. 4.1). It is assumed that the vehicle moves on a plane without

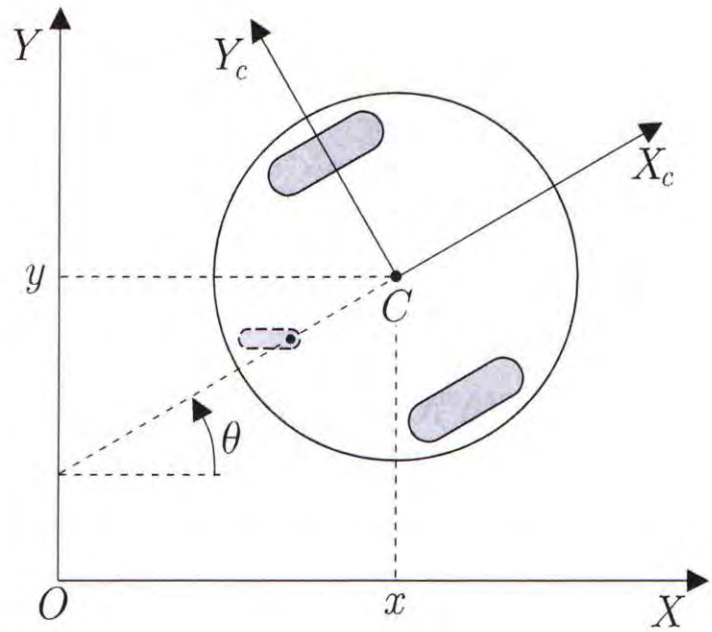

Figure 4.1: Coordinate system of the mobile robot

slipping, i.e., there is a pure rolling contact between the wheels and the ground. The kinematic model of the mobile robot is:

$$\begin{cases} \dot{x_c} = v\cos\theta, \\ \dot{y_c} = v\sin\theta, \\ \dot{\theta} = w, \end{cases} \tag{4.15}$$

where $[x_c, y_c]$ is the position of the robot center; θ is the orientation of the robot; v and w are the linear and the angular velocities, respectively.

By defining x=$[x_c \quad y_c \quad \theta]$ and $u = [v, w]$, the above equations can be rewritten as:

$$\dot{x} = f(x, u) \tag{4.16}$$

where x describe the position and orientation of the robot, u is the control input.

A discrete-time version of the system (4.16) can be obtained with a sampling period 0.1. The proposed RNN approach is applied to the mobile robot controller design.

4.3.1 Example 1

In this example, the initial position of the robot is $[0.25, 1]$. weighting matrices are $R = 5I$ and $Q = I$. Prediction horizon $N = 10$, control horizon $N_u = 5$. $\gamma = [0.001, 0.001, 0.01]$. The objective is to track a straight line. The mobile robot system is subject to the following constraints:

$$\begin{bmatrix} -1 \\ -1 \end{bmatrix} \leq u \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$
$$\begin{bmatrix} 0 \\ 0 \\ -2\pi \end{bmatrix} \leq x \leq \begin{bmatrix} 2.5 \\ 5 \\ 2\pi \end{bmatrix} \tag{4.17}$$

As shown in Fig. 4.2, the mobile robot can track the straight line precisely although its start point is outside the reference trajectory. The tracking mean square error (MSE) on X-axis and Y-axis are 0.0068 and 0.00731, respectively.

4.3.2 Example 2

In this example, the initial position of the robot is $[0, -0.5]$. weighting matrices are $R = I$ and $Q = 0.1I$. Prediction horizon $N = 10$, control horizon $N_u = 5$. $\gamma = [0.001, 0.001, 0.01]$. The objective is to track a square trajectory. The mobile robot system is subject to the following constraints:

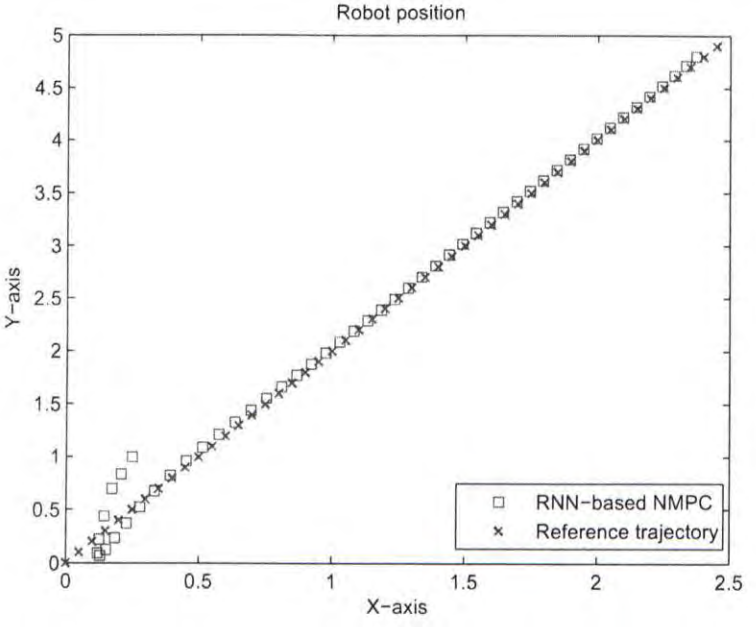

Figure 4.2: Trajectory

$$\begin{bmatrix} -1 \\ -1 \end{bmatrix} \leq u \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$
$$\begin{bmatrix} -0.5 \\ 0 \\ -2\pi \end{bmatrix} \leq x \leq \begin{bmatrix} 3 \\ 3 \\ 2\pi \end{bmatrix} \tag{4.18}$$

Simulation results are shown in Fig. 4.3, from which can be seen taht the robot can track the reference square trajectory precisely. The tracking MSE on X-axis and Y-axis are 0.0185 and 0.0199, respectively.

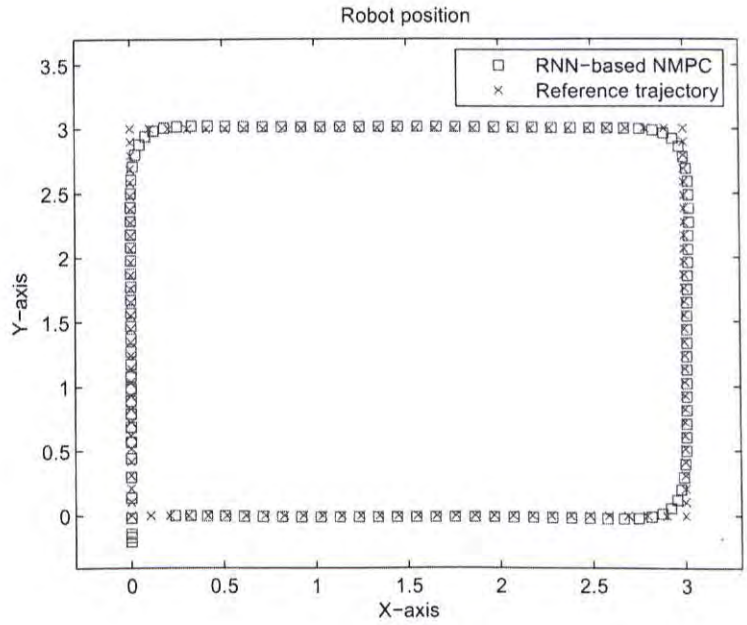

Figure 4.3: Trajectory

4.3.3 Example 3

In this example, the initial position of the robot is $[0, 0.15]$. weighting matrices are $R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$, $Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$. Prediction horizon $N = 10$, control horizon $N_u = 5$. $\gamma = [0.001, 0.001, 0.01]$. The objective is to track a curve line. The mobile robot system is subject to the following constraints:

$$\begin{bmatrix} -0.4 \\ -0.5 \end{bmatrix} \leq u \leq \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix}$$
$$\begin{bmatrix} -0.8 \\ 0 \\ -2\pi \end{bmatrix} \leq x \leq \begin{bmatrix} 1.1 \\ 1 \\ 2\pi \end{bmatrix} \tag{4.19}$$

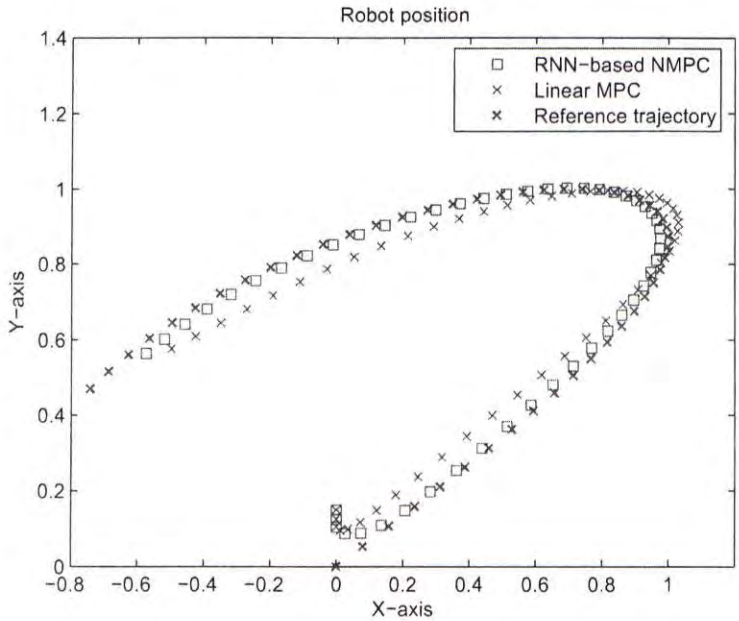

Figure 4.4: Trajectory

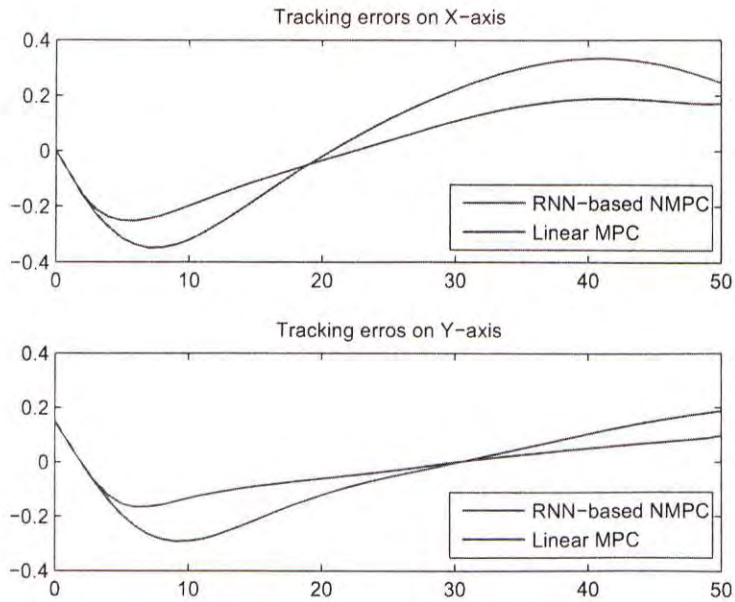

Figure 4.5: Tracking errors

In order to compare the effectiveness and efficiency of the proposed approach, linear MPC (discard the high-order term of Taylor series) is also applied to the system. The simulation results are compared in Fig. 4.4 and 4.5. The tracking mean square error (MSE) of the RNN-based scheme on X-axis and Y-axis are 0.0241 and 0.0066, respectively. While the MSE of linear MPC on X-axis and Y-axis are 0.0603 and 0.0119, respectively. As a result, the proposed RNN-based scheme gives better performance with less tracking errors.

4.3.4 Example 4

In this example, the initial position of the mobile robot is $[-1, 0]$. The weighting matrices are $R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$, $Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$. Prediction horizon $N = 10$, control horizon $N_u = 5$. $\gamma = [0.001, 0.001, 0.01]$. The mobile robot system is subject to the following constraints:

$$\begin{bmatrix} -0.4 \\ -0.5 \end{bmatrix} \leq u \leq \begin{bmatrix} 0.4 \\ 0.5 \end{bmatrix}$$
$$\begin{bmatrix} -2 \\ -1 \\ -2\pi \end{bmatrix} \leq x \leq \begin{bmatrix} 2 \\ 1 \\ 2\pi \end{bmatrix} \tag{4.20}$$

The control performances of the proposed scheme are also compared with linear MPC (discard the high-order term of Taylor series). The simulation results are shown in Fig. 4.6-4.7. The tracking MSE of the RNN-based scheme on X-axis and Y-axis are 0.0555 and 0.024, respectively. While the MSE of linear MPC on X-axis and Y-axis are 0.1103 and 0.0333, respectively. It can be seen that the proposed RNN-based NMPC scheme gives better tracking result with less errors.

□ **End of chapter.**

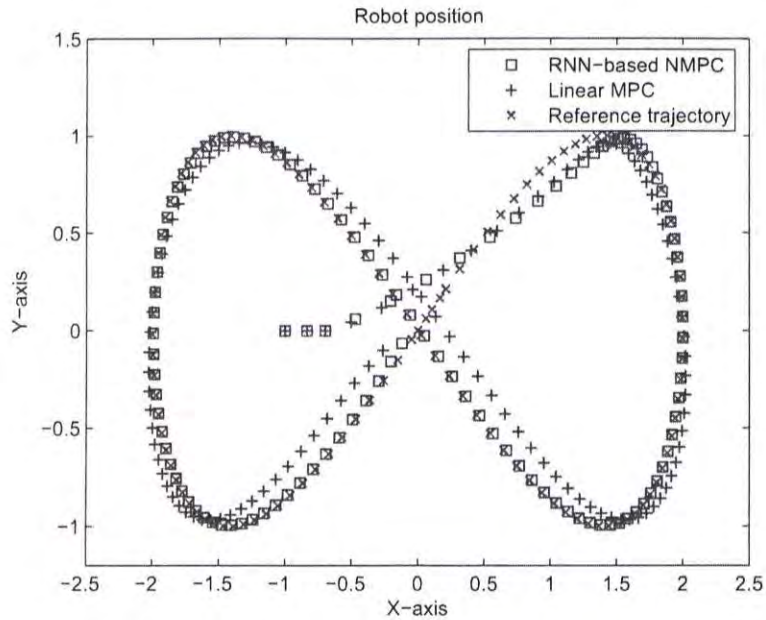

Figure 4.6: Trajectory

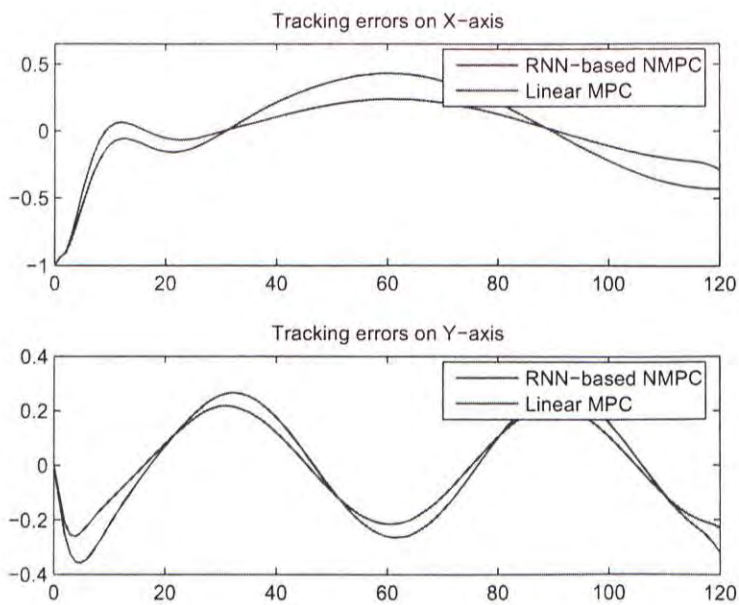

Figure 4.7: Tracking errors

Chapter 5

Model Predictive Control of Unknown Nonlinear Dynamic Systems Based on Recurrent Neural Networks

As many processes may be approximated locally using linear models within limited operating points, linear MPC was used in the majority of MPC applications with feedback compensation of prediction errors due to structural mismatch between the model and the process. As linear models are not sufficiently accurate because of process nonlinearity, standard MPC schemes based on linearization would result in poor performance. More challenging tasks on nonlinear MPC has also been attempted for nonlinear systems. In general, nonlinear MPC design based on nonlinear models results in formulations of nonconvex optimization problems. However, there is no effective approach can give global optimal solutions to such problems. Thus, further investigations on high-performance MPC for nonlinear systems are absolutely necessary.

In addition to online optimization, when the process model is unknown or un-

available, system identification is necessary. Numerous studies on identification of unknown nonlinear systems have been performed based on neural networks [68]-[71]. For system identification, multilayer feedforward networks have been used as universal approximators (e.g., [68], [69]). As the limitations of multilayer feedforward networks have become more apparent, RNNs have been shown to posses better capabilities and have been successfully applied for modeling and control of nonlinear dynamic systems. In recent years, a novel RNN called the echo state network (ESN) was introduced [72], which has showed good performance and require a very simple training. ESN has been successfully applied for system identification and control [73][74].

5.1 MPC System Description

Consider the following input-output representation of a general discrete-time nonlinear system:

$$\begin{aligned} y(k) = f(&y(k-1), \ldots, y(k-n_y), \\ &u(k-1), \ldots, u(k-n_u)), \end{aligned} \tag{5.1}$$

where $f(\cdot)$ is a unknown nonlinear function, $u(\cdot) \in \Re^m$ and $y(\cdot) \in \Re^n$ denote the process input and output, respectively, n_u and n_y are respectively the time delays of input and output.

5.1.1 Model Predictive Control

For system (5.1), the following cost function can be formulated and used for calculation of the optimal input trajectory over the control horizon:

$$J(k) = \sum_{j=1}^{N} \|r(k+j|k) - y(k+j|k)\|_Q^2 + \sum_{j=0}^{N_u-1} \|\Delta u(k+j|k)\|_R^2 \tag{5.2}$$

where $r(k+j|k)$ denotes the reference trajectory of output signal (desired output) at sampling instant $k(j \geq 0)$, $y(k+j|k)$ denotes the predicted output, $\Delta u(k+$

$j|k) = u(k+j|k) - u(k+j-1|k)$ denotes the input increment, N is the predictive horizon ($N \geq 1$), N_u denotes the control horizon ($N \geq N_u > 0$), $Q \in \Re^{n\times n}$ and $R \in \Re^{m\times m}$ are appropriate weighting matrices, and $\|\cdot\|_Q$ and $\|\cdot\|_R$ are weighted norms defined as $\|z\|_W = \sqrt{z^T W z}$.

The following bound constraints are considered:

$$\begin{aligned} u_{\min} \leq & u(k+j|k) \leq u_{\max}, \quad j = 0, 1, ..., N_u - 1, \\ -\Delta u_{\max} \leq & \Delta u(k+j|k) \leq \Delta u_{\max}, j = 0, 1, ..., N_u - 1, \\ y_{\min} \leq & y(k+j|k) \leq y_{\max}, \quad j = 1, 2, ..., N. \end{aligned} \tag{5.3}$$

where $\Delta u_{\max} > 0$, $u_{\min} \leq u_{\max}$, and $y_{\min} \leq y_{\max}$ are vectors of upper and lower bounds.

5.1.2 Dynamical System Identification

To identify the unknown system (5.1), we employ the echo state network.

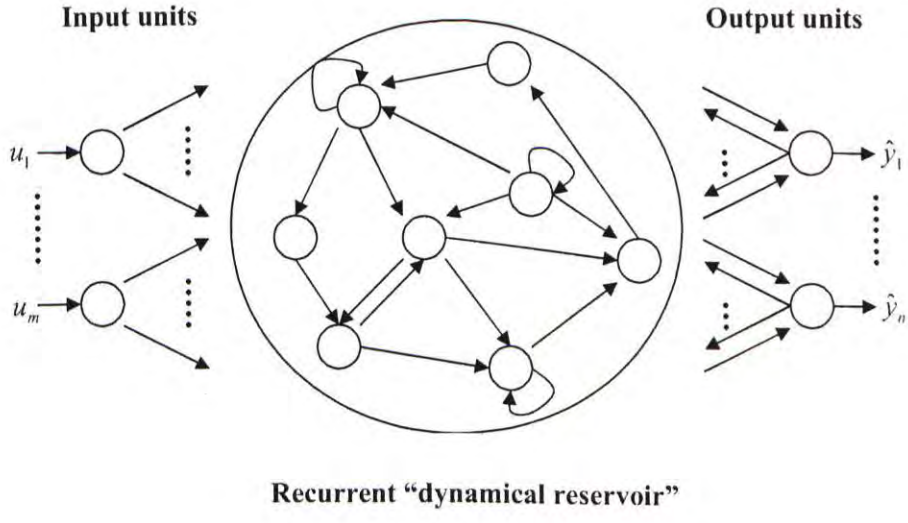

Figure 5.1: Architecture of an ESN

ESN Architecture

The echo state network (ESN), proposed by Jaeger [20], is a recurrent neural network for dynamic system modeling. It is composed of a hidden layer (dynamical reservoir) and an output layer (readout). Its structure is illustrated in Fig. 5.1. Here we consider an ESN with m inputs, r neurons in the hidden layer (reservoir), and n neurons in the output layer:

$$\begin{aligned} x(k+1) &= h(W_u u(k+1) + W_x x(k) + W_y \hat{y}(k)), \\ \hat{y}(k+1) &= W x(k+1), \end{aligned} \tag{5.4}$$

where $h(\cdot)$ is a vector-valued sigmoid activation function; $W_u \in \Re^{r\times m}$, $W_x \in \Re^{r\times r}$, $W_y \in \Re^{r\times n}$, and $W \in \Re^{n\times r}$ denote the input-internal, internal-internal, output-internal, and internal-output connection weight matrices; $x(\cdot) \in \Re^r$ and $\hat{y}(\cdot) \in \Re^n$ are the state and output vector of the ESN, respectively.

ESN Training

The training of an ESN is simple. The values of W_u, W_x, and W_y can be randomly chosen and fixed during training, only W will be updated. To compute W, the mean squared training error (MSE) is minimized:

$$MSE = \frac{1}{k_2 - k_1} \sum_{k_1}^{k_2} (y(k) - \hat{y}(k))^2 \tag{5.5}$$

where $y(k) \in \Re^n$ is the target sequence (output of the original system model (5.1)), $k_1 < k_2$, $k = k_1, \cdots, k_2$.

During the training of an ESN, the target sequence $y(k)$ and actual ESN output are compared and the errors are used to compute only the output weighting matrix (readout) W while all other weights in the ESN do not change. Thus, the training of the ESN reduces to the training of a feedforward network like an Adaline. As a result, the computational complexity for ESN training is low. In addition, The convergence of ESN learning is guaranteed [75], and it does not

suffer from local minima. In this paper, ESNs are adopted to identify unknown dynamic systems off-line. In the next section, we give the formulation of the MPC synthesis problem based on the ESN model (5.4).

5.2 Problem Formulation

Note that the optimization problem associated with MPC is nonconvex based on the ESN model (5.4), as $h(\cdot)$ is a nonlinear function. In this paper, we choose $h(s) = 1/(1+e^{-s})$. By using Taylor expansion, the ESN model can be decomposed into a known linear part plus an unknown nonlinear part around $s(k)$:

$$\begin{aligned}
x(k+1) =& h(s) \\
=& h(s(k)) + \nabla h(s(k))(s - s(k)) + \theta(s - s(k)) \\
=& \nabla h(s(k))s + h(s(k)) - \nabla h(s(k))s(k) + \theta(s - s(k)) \\
=& \nabla h(s(k))W_u u(k+1) + \nabla h(s(k))W_x x(k) + \nabla h(s(k)) \\
& W_y \hat{y}(k) + h(s(k)) - \nabla h(s(k))s(k) + \theta(\varepsilon(k)), \\
\hat{y}(k+1) =& Wx(k+1) \\
=& Au(k+1) + Bx(k) + C\hat{y}(k) + c + \theta(\varepsilon(k)),
\end{aligned}$$

where

$$\begin{aligned}
s(k) &= W_u u(k) + W_x x(k-1) + W_y \hat{y}(k-1), \\
A &= W\nabla h(s(k))W_u \in \Re^{n\times m}, \\
B &= W\nabla h(s(k))W_x \in \Re^{n\times r}, \\
C &= W\nabla h(s(k))W_y \in \Re^{n\times n}, \\
c &= h(s(k)) - \nabla h(s(k))s(k), \\
\nabla h(s(k)) &= \begin{bmatrix} \frac{e^{-s_1(k)}}{(1+e^{-s_1(k)})^2} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{e^{-s_r(k)}}{(1+e^{-s_r(k)})^2} \end{bmatrix} \in \Re^{r\times r},
\end{aligned}$$

$\nabla h(s(k))$ is the Jacobian matrix of $h(\cdot)$ at $s(k)$, $\theta(\varepsilon(k))$ is the unknown nonlinear term of the Taylor series where $\varepsilon(k) = s - s(k)$.

The predicted output increment vector can be computed as follows:

$$
\begin{aligned}
\Delta\hat{y}(k+1|k) =& \hat{y}(k+1|k) - \hat{y}(k|k) \\
=& A\Delta u(k+1|k) + B\Delta x(k|k) + C\Delta\hat{y}(k|k) \\
& + \Delta\theta(\varepsilon(k+1|k)), \\
\Delta\hat{y}(k+2|k) =& \hat{y}(k+2|k) - \hat{y}(k+1|k) \\
=& A\Delta u(k+2|k) + B\Delta x(k+1|k) + \\
& C\Delta\hat{y}(k+1|k) + C^2\Delta\hat{y}(k|k) + \\
& \Delta\theta(\varepsilon(k+2|k)) \\
=& A\Delta u(k+2|k) + A\Delta u(k+1|k) + \\
& B\Delta x(k|k) + B\Delta x(k+1|k) + C\Delta\hat{y}(k|k) + \\
& C^2\Delta\hat{y}(k|k) + \Delta\theta(\varepsilon(k+1|k)) + \\
& \Delta\theta(\varepsilon(k+2|k)), \\
& \vdots \\
\Delta\hat{y}(k+N|k) =& \hat{y}(k+N|k) - \hat{y}(k-1+N|k) \\
=& A\Delta u(k+N|k) + CA\Delta u(k+N-1|k) + \\
& \cdots + C^{N-1}A\Delta u(k+1|k) + \\
& B\Delta x(k+N-1|k) + CB\Delta x(k+N-2|k) \\
& + \cdots + C^{N-1}b\Delta x(k|k) + (C + C^2 + \cdots \\
& + C^N)\Delta\hat{y}(k|k) + \Delta\theta(\varepsilon(k+1|k)) \\
& + \Delta\theta(\varepsilon(k+2|k)) + \cdots + \\
& \Delta\theta(\varepsilon(k+N|k)).
\end{aligned}
$$

Define following vectors:

$$\begin{aligned}
\bar{y}(k) &= [\hat{y}(k+1|k)^T \quad \cdots \quad \hat{y}(k+N|k)^T]^T \in \Re^{Nn}, \\
\Delta\bar{y}(k) &= [\Delta y(k+1|k)^T \quad \cdots \quad \Delta y(k+N|k)^T]^T \in \Re^{Nn}, \\
\bar{x}(k) &= [x(k|k)^T \quad \cdots \quad x(k+N-1|k)^T]^T \in \Re^{Nr} \\
\Delta\bar{u}(k) &= [\Delta u(k|k)^T \quad \cdots \quad \Delta u(k+N_u-1|k)^T]^T \in \Re^{N_u m}, \\
\bar{r}(k) &= [r(k+1|k)^T \quad \cdots \quad r(k+N|k)^T]^T \in \Re^{Nn}, \\
\Delta\bar{\theta}(\varepsilon(k)) &= [\Delta\theta(\varepsilon(k+1|k))^T \quad \cdots \Delta\theta(\varepsilon(k+N|k))^T] \in \Re^{Nn},
\end{aligned}$$

where $\Delta\bar{\theta}(\varepsilon(k))$ is an unknown vector, $\bar{r}(k)$ is a given reference trajectory. Then the predicted output $\bar{y}(k)$ can be expressed as follows:

$$\begin{aligned}
\bar{y}(k) =& \bar{y}(k-1) + \Delta\bar{y}(k) \\
=& \bar{y}(k-1) + \tilde{A}\Delta\bar{u}(k) + \tilde{B}\Delta\bar{x}(k) + \tilde{C}\Delta y(k|k) \\
&+ \Delta\bar{\theta}(\varepsilon(k)),
\end{aligned} \tag{5.6}$$

where

$$\tilde{A} = \begin{bmatrix} A & 0 & \cdots & 0 \\ CA & A & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C^{N-1}A & C^{N-2}A & \cdots & A \end{bmatrix} \in \Re^{Nn\times Nm},$$

$$\tilde{B} = \begin{bmatrix} B & 0 & \cdots & 0 \\ CB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C^{N-1}B & C^{N-2}B & \cdots & B \end{bmatrix} \in \Re^{Nn\times Nr},$$

$$\tilde{C} = [C \quad C^2 \quad \cdots \quad C^N]^T \in \Re^{Nn\times n}.$$

The original optimization problem (5.2) and (5.3) associated with MPC can be

rewritten as follows:

$$\begin{aligned} \min \quad & \|\bar{y}(k) - \bar{r}(k)\|_Q^2 + \|\Delta\bar{u}(k)\|_R^2 \\ \text{s.t.} \quad & -\Delta\bar{u}_{\max} \leq \Delta\bar{u}(k) \leq \Delta\bar{u}_{\max}, \\ & \bar{u}_{\min} \leq \bar{u}(k-1) + H\Delta\bar{u}(k) \leq \bar{u}_{\max}, \\ & \bar{y}_{\min} \leq \bar{y}(k) \leq \bar{y}_{\max}, \end{aligned} \tag{5.7}$$

where

$$\begin{aligned} \Delta\bar{u}_{\max} &= [\Delta u_{\max}^T \quad \cdots \quad \Delta u_{\max}^T]^T \in \Re^{N_u m}, \\ \bar{u}_{\min} &= [u_{\min}^T \quad \cdots \quad u_{\min}^T]^T \in \Re^{N_u m}, \\ \bar{u}_{\max} &= [u_{\max}^T \quad \cdots \quad u_{\max}^T]^T \in \Re^{N_u m}, \\ \bar{y}_{\min} &= [y_{\min}^T \quad \cdots \quad y_{\min}^T]^T \in \Re^{Nn}, \\ \bar{y}_{\max} &= [y_{\max}^T \quad \cdots \quad y_{\max}^T]^T \in \Re^{Nn}, \\ H &= \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & I & I \end{bmatrix} \in \Re^{N_u m \times N_u m}. \end{aligned}$$

The quadratic optimization problem (5.7) can be rewritten in the following concise form:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\Delta\bar{u}(k)\|_W^2 + p^T\Delta\bar{u}(k) \\ \text{s.t.} \quad & q_{\min} \leq E\Delta\bar{u}(k) \leq q_{\max}, \end{aligned} \tag{5.8}$$

where

$$
\begin{aligned}
W =& 2(\tilde{A}^T Q \tilde{A} + R) \in \Re^{N_u m \times N_u m}, \\
p =& 2\tilde{A}^T Q(\bar{y}(k-1) + \tilde{B}\Delta\bar{x}(k) + \tilde{C}\Delta y(k|k) + \Delta\bar{\theta}(\varepsilon(k)) \\
& - \bar{r}(k)) \in \Re^{N_u m}, \\
E =& [I \quad H \quad \tilde{A}]^T \in \Re^{(2N_u m + Nn) \times N_u m},
\end{aligned}
$$

$$
q_{\min} = \begin{bmatrix} -\Delta\bar{u}_{\max} \\ \bar{u}_{\min} - \bar{u}(k-1) \\ \bar{y}_{\min} - \bar{y}(k-1) - \tilde{B}\Delta\bar{x}(k) - \tilde{C}\Delta y(k|k) - \Delta\bar{\theta}(\varepsilon(k)) \end{bmatrix},
$$

$$
q_{\max} = \begin{bmatrix} \Delta\bar{u}_{\max} \\ \bar{u}_{\max} - \bar{u}(k-1) \\ \bar{y}_{\max} - \bar{y}(k-1) - \tilde{B}\Delta\bar{x}(k) - \tilde{C}\Delta y(k|k) - \Delta\bar{\theta}(\varepsilon(k)) \end{bmatrix}.
$$

If Q and R are positive definite, then W is positive definite, which means the objective function of (5.8) is strictly convex (due to W being positive definite), and the feasible region of linear constraints is a convex set, so the solution to the QP problem (5.8) is unique and satisfies the Karush-Kahn-Tucker (KKT) optimality conditions.

The solution to (5.8) gives the vector of control action $\Delta\bar{u}(k)$, which is used to calculate the optimal input vector. It should be noticed that the vector $\bar{\theta}(\varepsilon(k))$ is unknown, thus, the vectors p, $q_{\min}$, and $q_{\max}$ are unknown. In the next section, we propose an RNN approach to solving this problem.

5.3 Dynamic Optimization

In this section, we employ the simplified dual neural network [28] for controller design.

5.3.1 The Simplified Dual Neural Network

By considering (4.8) as the primal problem, its dual problem is:

$$\begin{aligned} \max \quad & -\frac{1}{2}\|\Delta\bar{u}(k)\|_W^2 + q_{\min}^T v - q_{\max}^T \omega \\ \text{s.t.} \quad & W\Delta\bar{u}(k) + p - E^T q_{\min} + E^T q_{\max} = 0 \end{aligned} \tag{5.9}$$

where $v \in \Re^{2N_u m+Nn}$ and $\omega \in \Re^{2N_u m+Nn}$ are dual decision variables. Define the variable vector $v = \Delta\bar{u}(k)$, $w = v - \omega$. According to the KKT conditions, the following equations have the same solution as the primal problem (5.8) and its dual (5.9):

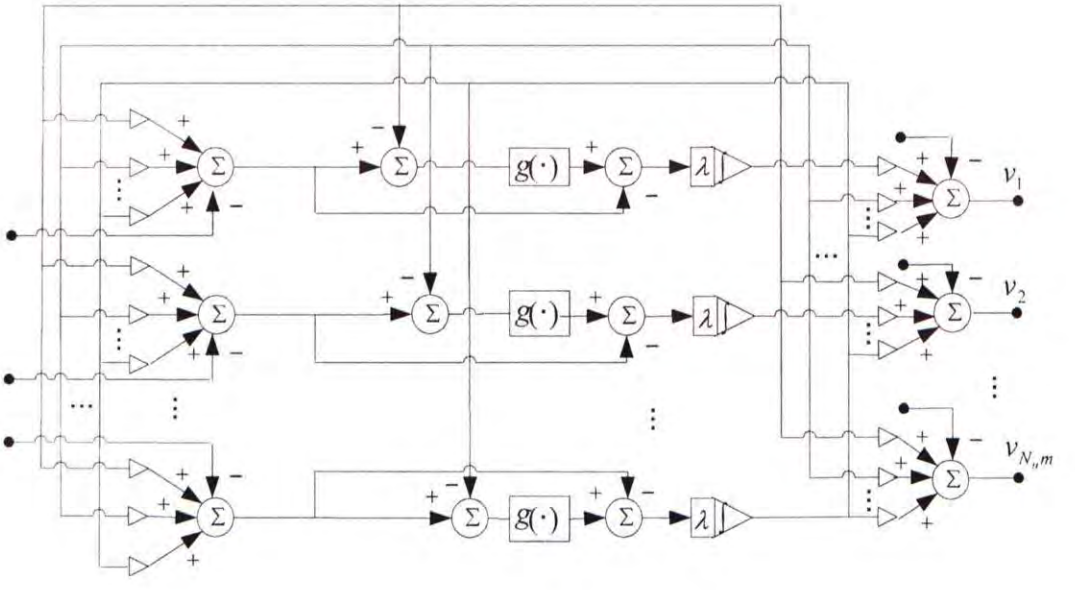

Figure 5.2: Architecture of the simplified dual neural network

$$\begin{aligned} & Wv + p - E^T w = 0, \\ & Ev = g(Ev - w), \end{aligned} \tag{5.10}$$

where $g(\cdot)$ is a piecewise linear activation function, defined as:

$$g(\mu) = \begin{cases} q_{\min}, & \mu < q_{\min}; \\ \mu, & q_{\min} \leq \mu \leq q_{\max}; \\ q_{\max}, & \mu > q_{\max}. \end{cases} \tag{5.11}$$

Based on (5.10) and (5.11), the dynamic equation of the simplified dual neural network for solving (5.8) can be designed as [9]:

- State equation

$$\begin{aligned} \frac{dw}{dt} =& \lambda(-EW^{-1}E^T w + g(EW^{-1}E^T w - EW^{-1}p - w) \\ &+ EW^{-1}p), \end{aligned} \tag{5.12}$$

- Output equation

$$v = W^{-1}E^T w - W^{-1}p, \tag{5.13}$$

where $\lambda > 0$ is a scaling constant, $w \in \Re^{2N_u m + Nn}$ is the state vector.

The above RNN model can be implemented with a single-layer structure of $2N_u m + Nn$ neurons. Its structure is shown in Fig. 5.2. According to the convergence analysis in [9], the simplified dual network is Lyapunov stable and globally convergent to the optimal solution to (5.8). However, the nonlinear term of Taylor series $\Delta\bar{\theta}(\varepsilon(k))$ is still unknown. Thus, the RNN parameters p, $q_{\min}$, and $q_{\max}$ that contain $\Delta\bar{\theta}(\varepsilon(k))$ are unknown. As a result, the RNN (5.12) and (5.13) can not be applied in MPC design directly and a learning algorithm for estimating unknown parameters is necessary.

5.3.2 A Recursive Learning Algorithm

In this subsection, we propose a recursive learning algorithm to estimate the unknown parameters. For brevity, denote $\alpha(k) = \alpha_k$:

1. Initialization: Set $\Delta\bar{\theta}(\varepsilon_k(0)) = 0$, calculate $p(0)$, $q_{\min}(0)$, $q_{\max}(0)$, W, and E.

2. Calculate $\Delta\bar{\theta}(\varepsilon_k(i+1))$ as:

$$\Delta\bar{\theta}(\varepsilon_k(i+1)) = \Delta\bar{\theta}(\varepsilon_k(i)) + \delta(i)(\hat{y}_k(i) - \bar{y}_k(i)), \tag{5.14}$$

where $\delta(i)$ is a positive and decreasing convergence factor $\delta(i) < 1$ (choose $\delta(i) = 1/i$ in this paper), $\hat{y}_k$ is the output of ESN (5.4).

3. Update RNN parameters $p(i+1)$, $q_{\min}(i+1)$, and $q_{\max}(i+1)$ with $\Delta\bar{\theta}(\varepsilon_k(i+1))$.

4. Solve the QP problem (4.8) using the proposed neural network model (5.12) and (5.13) to obtain the control action $\Delta\bar{u}_k(i+1)$.

5. If $|\Delta\bar{\theta}(\varepsilon_k(i+1)) - \Delta\bar{\theta}(\epsilon_k(i))| < \gamma$ for some small positive γ, stop; otherwise, $i = i+1$, return to Step 2 for another iteration.

The basic idea of this learning algorithm is to estimate the unknown nonlinear term of Taylor series by recursive calculation. By properly choosing $\delta(i)$ and γ, the learning algorithm will enable the RNN to give near-optimal solutions to the optimization problem (5.8).

5.3.3 Convergence Analysis

Now we prove the convergence property of the proposed RNN optimization approach.

Lemma 1 [28]: The output trajectory of the simplified dual neural network is globally convergent to a unique optimal solution of (5.8), if Q and R are symmetric and positive definite matrices for fixed $p(i)$, $q_{\min}(i)$, and $q_{\max}(i)$ $(i = 0, 1, 2 \ldots)$.

Lemma 2 [33]: Consider the following equations:

$$\begin{aligned} \beta(i) &= \beta(i-1) + \delta(i)\kappa(i, \beta(i-1), \xi(i, \beta)), \\ \xi(i) &= \vartheta(\beta(i-1))\xi(i-1, \beta) + \varsigma(\beta(i-1))\zeta(i) \end{aligned} \tag{5.15}$$

where $\beta(i)$ is the variable to be updated. Let $D=\{\beta|\vartheta(\beta)$ has all eigenvalues inside the unit circle$\}$, $\hat{D}$ is a connected open subset of D. In $\hat{D}$, the functions in (5.15) satisfy Conditions C_1-C_6:

C_1: The function $\kappa(i,\beta,\xi)$ is Lipschitz continuous in β and ξ in any neighborhood of $(\hat{\beta},\hat{\xi})$, where $\hat{\beta}\in\hat{D}$ and $\hat{\xi}$ is arbitrary; $\kappa(i,\beta,\xi)$ is a continuous differentiable function of β and ξ.

C_2: The functions $\vartheta(\beta)$ and $\varsigma(\beta)$ are Lipschitz continuous in β for $\beta\in\hat{D}$.

C_3: $\xi(i,\hat{\beta})$ is defined as:

$$\xi(i,\hat{\beta})=\vartheta(\hat{\beta})\xi(i-1,\hat{\beta})+\varsigma(\hat{\beta})\zeta(i),\quad \xi(0,\hat{\beta})=0.$$

For all $\hat{\beta}\in\hat{D}$ where $\varrho(i,j)=\delta(j)\prod_j^i[1-\delta(j+1)]$,

$$\sum_{j=1}^{i}\varrho(i,j)\kappa(j,\hat{\beta},\xi(j,\hat{\beta}))\to\alpha(\hat{\beta})\quad as\quad i\to\infty.$$

C_4: For all $\beta\in\hat{D}$ we have, for some $C^*(\hat{\beta})$ (assume that there is a constant $C^*<\infty$),

$$\sum_{j=1}^{i}\varrho(i,j)[1+\nu(j,\lambda^*,c^*)]\cdot K(\hat{\beta},\xi(j,\hat{\beta}),\rho(\hat{\beta}),\nu(j,\lambda^*,c^*))$$

$\to C^*(\hat{\beta})<\infty\quad as\quad i\to\infty$, where λ^* is the maximum eigenvalue of $\vartheta(\hat{\beta})$, $\nu(i,\lambda^*,c^*)$ is defined by:

$$\nu(i,\lambda^*,c^*)=c^*\sum_{j=1}^{i}\lambda^{*(i-j)}|\zeta(j)|,$$

and K is a Lipschitz constant.

C_5: $\sum_{i=1}^{\infty}\delta(i)=\infty$.

C_6: $\delta(i)\to 0$ as $i\to\infty$.

The corresponding differential equation of (5.15) is

$$\frac{d}{d\tau}\beta(\tau) = f^*(\beta(\tau)) \tag{5.16}$$

where $\tau = \Sigma_1^i \delta(i)$, and $f^* = \lim_{i\to\infty} \vartheta(\cdot)$.

If a Lyapunov function of (5.16) $V(\beta)$ exists, with

$$\frac{d}{d\tau}V(\beta) \leq 0, \quad \forall \beta \in \hat{D}, \tag{5.17}$$

then

$$\beta(i) \to \{\beta | \beta \in \hat{D}, \frac{d}{d\tau}V(\beta) = 0\} \quad as \quad i \to \infty,$$

if β^* is an asymptotically stable equilibrium point, $\beta(i) \to \beta^*$ as $i \to \infty$.

Theorem 1: At each time interval k, the output of the simplified dual network is globally convergent when $i \to \infty$, if $\delta(i) = 1/i$.

Proof: For brevity, at k, denote $\Delta\bar{\theta}(\varepsilon(k))$ with $\Delta\bar{\theta}$. According to (5.6) and (5.14), $\Delta\bar{\theta}(i+1)$ can be expressed as:

$$\begin{aligned}\Delta\bar{\theta}(i+1) =& \Delta\bar{\theta}(i) + \delta(i)(\hat{y}(i) - \bar{y}(i)) \\ =& \Delta\bar{\theta}(i) + \delta(i)(\hat{y}_k - \bar{y}_{k-1} - A\Delta\bar{u}_k(i) + B\Delta\bar{x}_k \\ & - C\Delta y_{k|k} - \Delta\bar{\theta}(i)) \\ =& \tilde{A}\Delta\bar{\theta}(i) + \tilde{B},\end{aligned} \tag{5.18}$$

where

$$\tilde{A} = \begin{bmatrix} 1-\delta(i) & 0 & \cdots & 0 \\ 0 & 1-\delta(i) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1-\delta(i) \end{bmatrix}, \tag{5.19}$$

$$\tilde{B} = \delta(i)(\hat{y}_k - y_{k-1} - A\Delta\bar{u}_k(i) + B\Delta\bar{x}_k - C\Delta y_{k|k}).$$

Define D=$\{\tilde{A}|\ \tilde{A}$ has all eigenvalues inside the unit circle$\}$. As

$$\lambda(\tilde{A}) = 1 - \delta(i) < 1, \tag{5.20}$$

then $D = \Re$. Conditions C_1 and C_2 are satisfied.

For C_3, we find that if $\delta(j) = 1/j$, then

$$\begin{aligned}\varrho(i,j) =& \delta(j)\prod_{j}^{i}[1-\delta(j+1)] \\ =& \frac{1}{j}\cdot\frac{j}{j+1}\cdots\frac{i-1}{i}\cdot\frac{i}{i+1} \\ =& \frac{1}{i+1}.\end{aligned} \tag{5.21}$$

As

$$\begin{aligned}\kappa(j,\Delta\bar{\theta},\xi(j,\Delta\bar{\theta})) &= \xi(j,\Delta\bar{\theta}) - \Delta\bar{\theta} \\ &= \hat{y}_k - \bar{y}_{k-1} - A\Delta\bar{u}_k(j) + B\Delta\bar{x}_k - C\Delta y_{k|k} - \Delta\bar{\theta},\end{aligned}$$

C_3 is satisfied.

To verify C_4, we note that as $\vartheta(\beta) = 0$, its maximum eigenvalue λ^* is also 0. Then $\nu(i,\lambda^*,c^*) = 0$, by choosing $K = 1$, condition C_4 becomes:

$$\sum_{j=1}^{i}\varrho(i,j) \to C^* \quad as \quad i\to\infty.$$

As

$$\sum_{j=1}^{i}\varrho(i,j) = \sum_{j=1}^{i}\frac{i}{i+1} \to 1 \quad as \quad i\to\infty,$$

then C_4 is satisfied with $C^* = 1$.

Both condition C_5 and C_6 are satisfied for $\delta(i) = 1/i$.

The corresponding differential equation of (5.18) is

$$\begin{aligned}\frac{d\Delta\bar{\theta}}{d\tau} &= f^*(\Delta\bar{\theta}) \\ &= \lim_{i\to\infty}(\hat{y}_k - \bar{y}_{k-1} - A\Delta\bar{u}(i)_k + B\Delta\bar{x}_k - C\Delta y_{k|k} - \\ &\quad \Delta\bar{\theta}(i)) \\ &= \Delta\check{\theta} - \Delta\bar{\theta},\end{aligned} \tag{5.22}$$

where $\Delta\check{\theta}$ is the true value of $\Delta\bar{\theta}$.

Define $\Delta\bar{\theta}^*$ as the equilibrium point of (5.22), then

$$\Delta\bar{\theta}^* = \Delta\check{\theta}.$$

Consider a Lyapunov function of (5.22)

$$V(\Delta\bar{\theta}) = \frac{1}{2}\|\Delta\check{\theta} - \Delta\bar{\theta}\|^2 > 0, \tag{5.23}$$

then

$$\begin{aligned} \frac{d}{d\tau}V(\Delta\bar{\theta}) &= (\Delta\check{\theta} - \Delta\bar{\theta})^T \frac{d}{d\tau}(\Delta\check{\theta} - \Delta\bar{\theta}) \\ &= -(\Delta\check{\theta} - \Delta\bar{\theta})^T(\Delta\check{\theta} - \Delta\bar{\theta}) \\ &\leq 0, \quad \forall\Delta\bar{\theta}, \end{aligned} \tag{5.24}$$

where

$$\frac{d}{d\tau}V(\Delta\bar{\theta}) = 0, \quad \Delta\bar{\theta} = \Delta\bar{\theta}^* = \Delta\check{\theta}. \tag{5.25}$$

According to *Lemma 2*, $\Delta\bar{\theta} \to \Delta\check{\theta}$ as $i \to \infty$. According to *Lemma 1*, the output of the simplified dual network is globally convergent at each time interval k.

5.4 RNN-based MPC Scheme

The RNN-based nonlinear MPC scheme (a block diagram is shown in Fig. 5.3, with z^{-1} being the one-step backward shift operator) can be summarized as follows:

1. Let $k = 1$. Set terminal time T, sample time t, predictive horizon N, control horizon N_u, weighting matrices Q and R.

2. Identify the unknown system off-line using an echo state network.

3. Calculate process model matrices and neural network parameters (initial).

4. Obtain a control action $\Delta\bar{u}(k)$ using the simplified dual network and the proposed learning algorithm in Section IV-B.

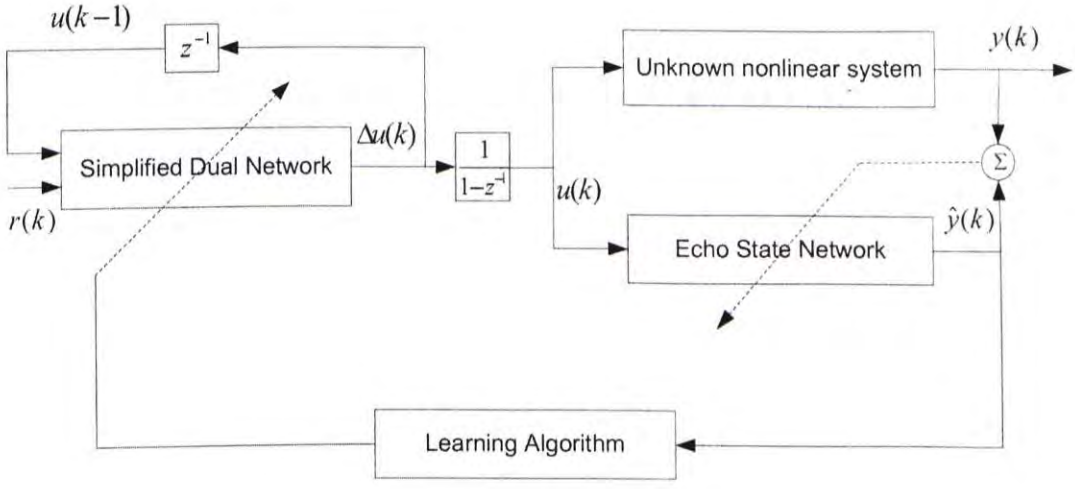

Figure 5.3: Block diagram of the RNN-based MPC scheme

5. Calculate and apply the control input $u(k) = \Delta u(k|k) + u(k-1)$.

6. Set $k = k + 1$, return to step 3.

The proposed RNN-based MPC scheme operates in a massively parallel fashion, which is suitable for large-scale system implementation.

5.5 Simulation Results

In this section, simulation results in three numerical examples are provided to illustrate the performance of the proposed RNN-based nonlinear MPC scheme.

5.5.1 Example 1

Consider the following nonlinear system with delay:

$$\begin{aligned} y(k) =& 0.2\sin(0.5(y(k-1)+y(k-2))) + 0.2\sin(0.5(y(k-2) \\ &+ y(k-3)) + 2u(k-1) + u(k-2)) \\ &+ \frac{4u(k-1)+u(k-2)}{1+0.2\cos(0.2(2y(k-1)+y(k-2))}, \end{aligned} \tag{5.26}$$

subject to input and output bound constraints:

$$-1.01 \leq y \leq 2.02, \quad -1 \leq u \leq 1.5.$$

The objective is to control the system to track a step change by applying the proposed RNN-based MPC scheme. For system identification, an ESN with 100 internal units is employed. The training results of ESN are shown in Fig. 5.4. The RNN-based controller parameters are chosen as: prediction horizon $N = 15$, control horizon $N_u = 10$, weighting matrices $Q = R = 5I$, $\gamma = 0.01$, sampling frequency is 10Hz.

In order to compare the effectiveness and efficiency of the proposed approach, linear MPC (based on linearization of the original model (5.26) without system

identification) is applied to the system. The simulation results are illustrated in Figs. 5.5 and 5.6. We can see that the proposed scheme gives a much better performance with more accurate set-point tracking.

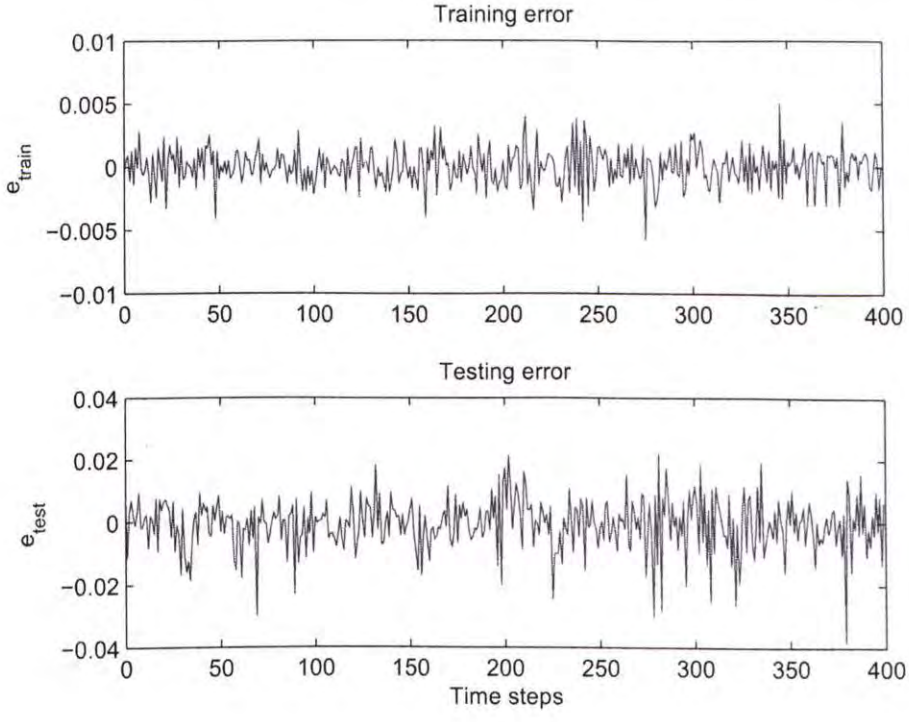

Figure 5.4: ESN training and testing errors in Example 1

5.5.2 Example 2

Consider a control system for a polymerization reactor described in [76], which has a strong nonlinear process model:

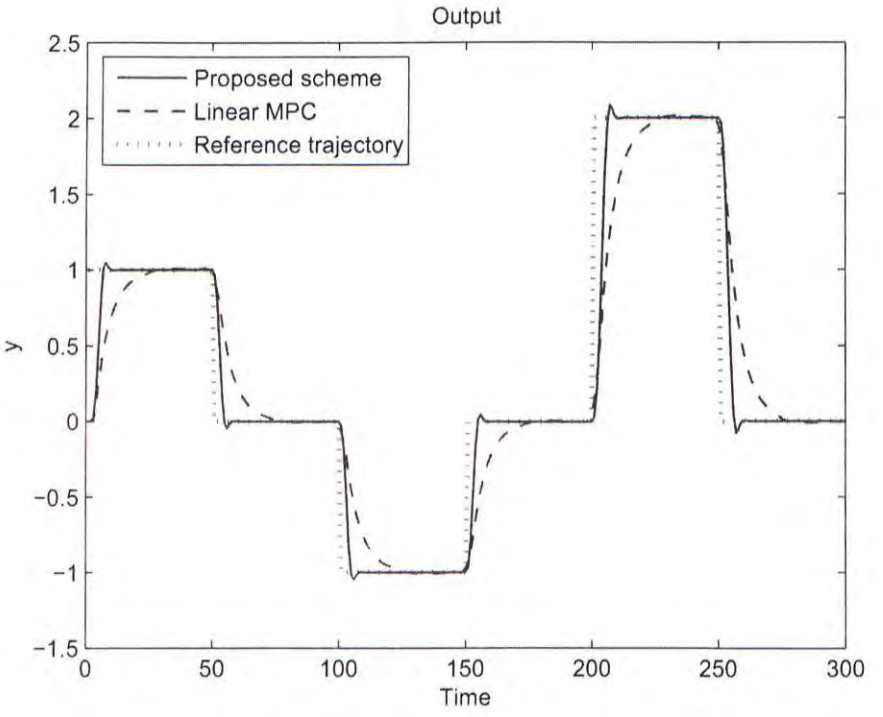

Figure 5.5: Output in Example 1

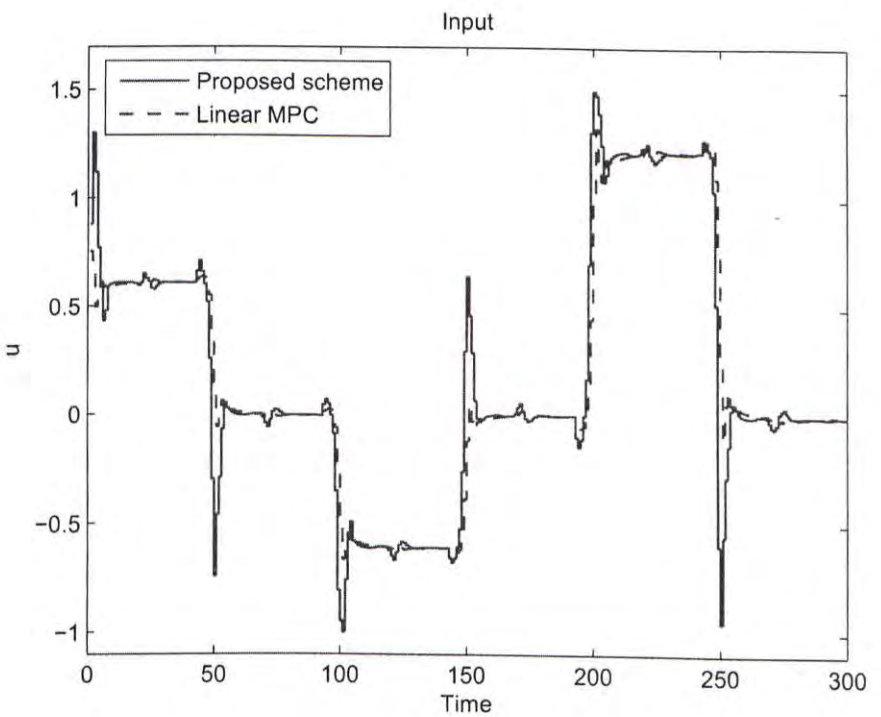

Figure 5.6: Control signal in Example 1

$$
\begin{aligned}
\dot{x}_1 =& 10(6 - x_1) - 2.4568x_1\sqrt{x_2}, \\
\dot{x}_2 =& 80u - 10.1022x_2, \\
\dot{x}_3 =& 0.002412x_1\sqrt{x_2} + 0.112191x_2 - 10x_3, \\
\dot{x}_4 =& 245.978x_1\sqrt{x_2} - 10x_4, \\
y =& \frac{x_4}{x_3},
\end{aligned}
\tag{5.27}
$$

subject to

$$0 \leq y \leq 36000, \quad 0.004 \leq u \leq 0.017,$$

where x_1 is the monomer concentration, x_2 is the initiator concentration, $y = x_4/x_3$ is the number-average molecular weight. The control problem is to regulate the number average molecular weight y by manipulating the initiator flow rate u. An ESN with 300 internal units is used for system identification. The training results are shown in Fig. 5.7. We notice that the identification precision in Example 2 is higher than Example 1 in terms of less errors. This is because the internal units of the ESN adopted in Example 2 is larger. Generally, a larger ESN would have a more precise learning result.

The RNN controller parameters are chosen as: prediction horizon $N = 10$, control horizon $N_u = 5$, sampling period 2s, weighting matrices $Q = I$, $R = 500I$, $\gamma = 10$. The system is discretized using zero-order holder. To compare the tracking performance, linear MPC (based on the linearized model of (5.27) without system identification) and RNN-based MPC scheme in [50] are both applied to the process model. Simulation results are shown in Figs. 5.8-5.10. As shown in Figs. 5.8 and 5.9, the proposed scheme gives the best performance with fastest and most stable tracking compared with the other two approaches. Fig. 5.10 shows the convergence behaviors of unknown nonlinear terms in different k.

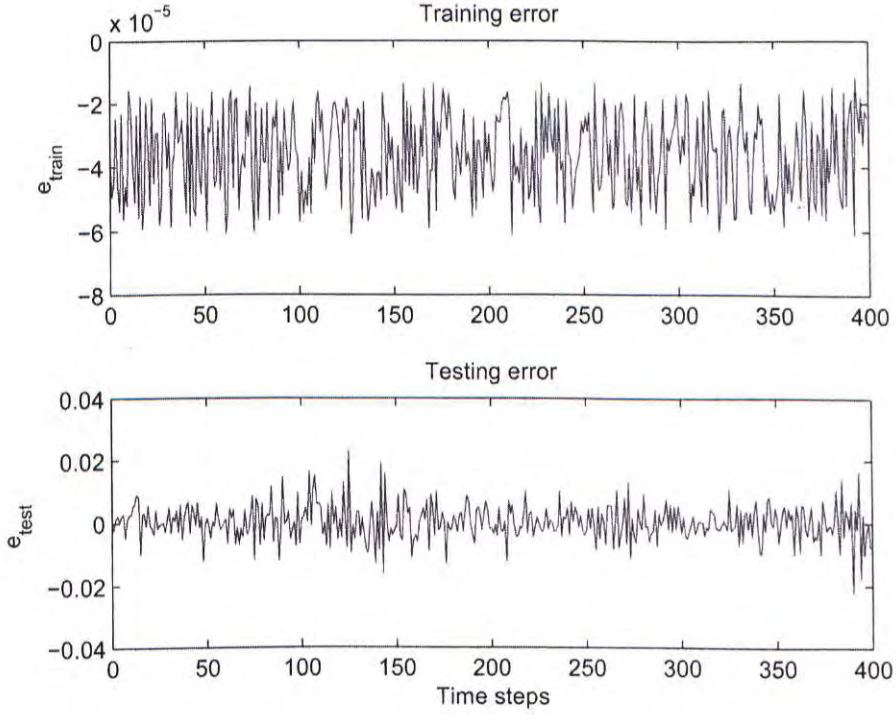

Figure 5.7: ESN training and testing errors in Example 2

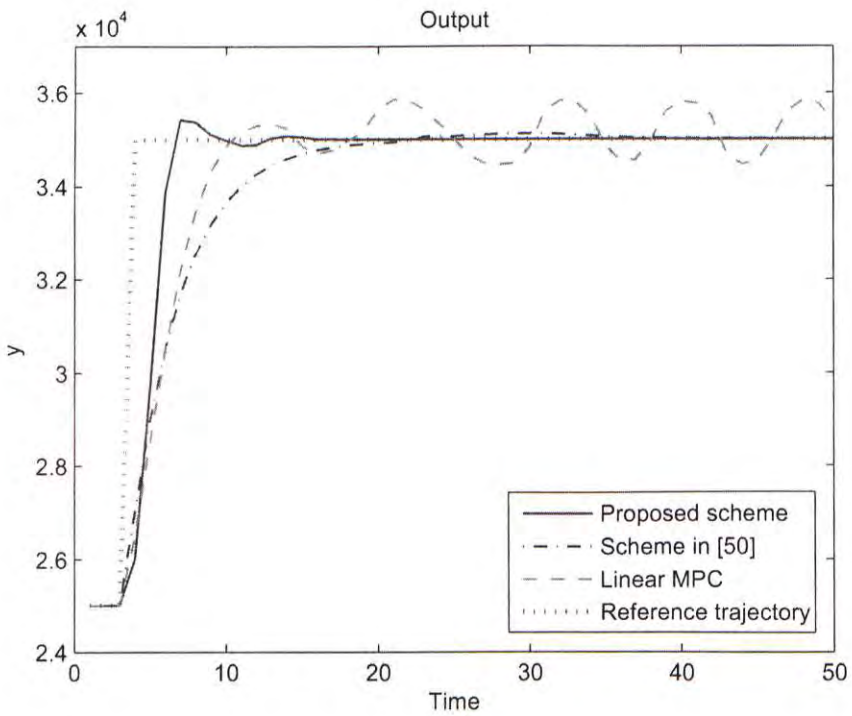

Figure 5.8: Output in Example 2

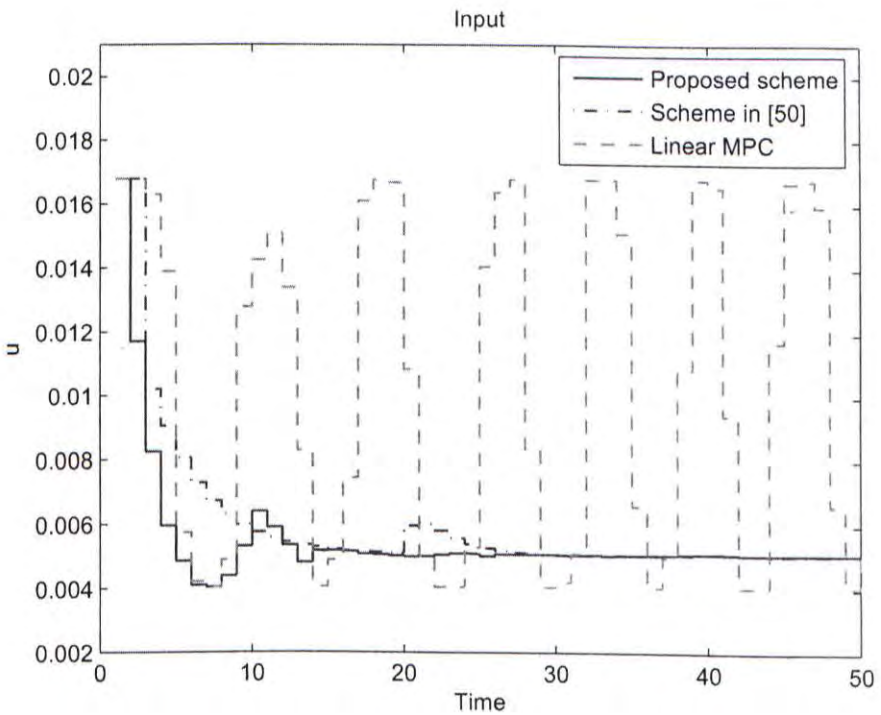

Figure 5.9: Control signal in Example 2

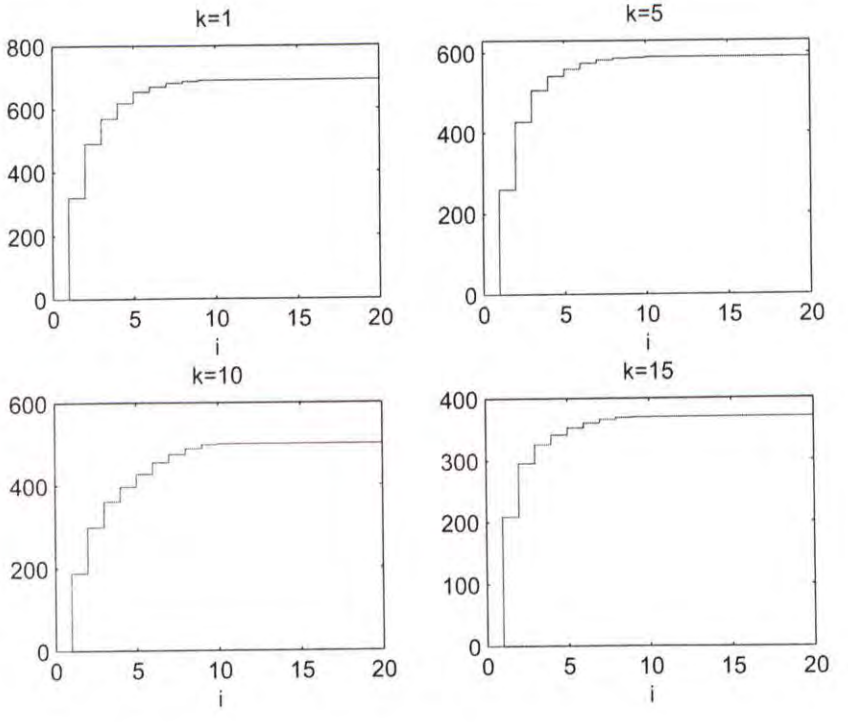

Figure 5.10: Convergence of the estimated unknown nonlinear terms in Example 2

5.5.3 Example 3

Consider the following multi-input multi-output (MIMO) nonlinear system:

$$\begin{aligned}
x_1(k) =& \frac{x_1^2(k-1)}{x_1^2(k-1)+1} + 0.5x_2(k-1) \\
x_2(k) =& \frac{x_1^2(k-1)}{x_2^2(k-1)+x_3^2(k-1)+x_4^2(k-1)} u_1(k-1) \\
x_3(k) =& \frac{x_3^2(k-1)}{x_3^2(k-1)+1} + 0.3x_4(k-1) \\
x_4(k) =& \frac{x_3^2(k-1)}{x_1^2(k-1)+x_2^2(k-1)+x_4^2(k-1)} + 0.5u_2(k-1) \\
y_1(k) =& x_1(k) + d(k), \quad y_2(k) = x_3(k) + d(k),
\end{aligned} \tag{5.28}$$

subject to:

$$\begin{aligned}
-1.3 \leq & y_1 \leq 0.7, \quad -1.5 \leq u_1 \leq 1.5, \\
-0.9 \leq & y_2 \leq 0.9, \quad -1.5 \leq u_2 \leq 1.5.
\end{aligned} \tag{5.29}$$

where $d(k)$ is a disturbance. The objective is to control the system to track the following reference trajectories:

$$\begin{aligned}
r_1 =& 0.75\sin(\frac{\pi k}{8}) + 0.5\cos(\frac{\pi k}{4}), \\
r_2 =& 0.5\cos(\frac{\pi k}{8}) + 0.5\cos(\frac{\pi k}{4}).
\end{aligned} \tag{5.30}$$

An ESN with 300 internal units is adopted for system identification. The training results are shown in Fig. 5.11. The system starts from initial states $[x_1(0), x_2(0), x_3(0), x_4(0)] = [0, 0, 0, 0]$. The parameters of the RNN-based controller are: prediction horizon $N = 15$, control horizon $N_u = 5$, sampling period $t = 1$s, weighting matrices $Q = R = I$, $\gamma = 1 \times 10^3$. The control performances of the proposed scheme are also compared with linear MPC (based on the linearized model of (5.28) without system identification). A step disturbance

$$d(k) = \begin{cases} -0.5, & 100 \leq k \leq 130; \\ 0, & 0 < k < 100 \quad \text{or} \quad 130 < k < 200, \end{cases} \tag{5.31}$$

is introduced to the system. Simulation results are shown in Figs. 5.12-5.14. As shown in Fig. 5.13, the proposed RNN-based controller gives better performances with less tracking errors.

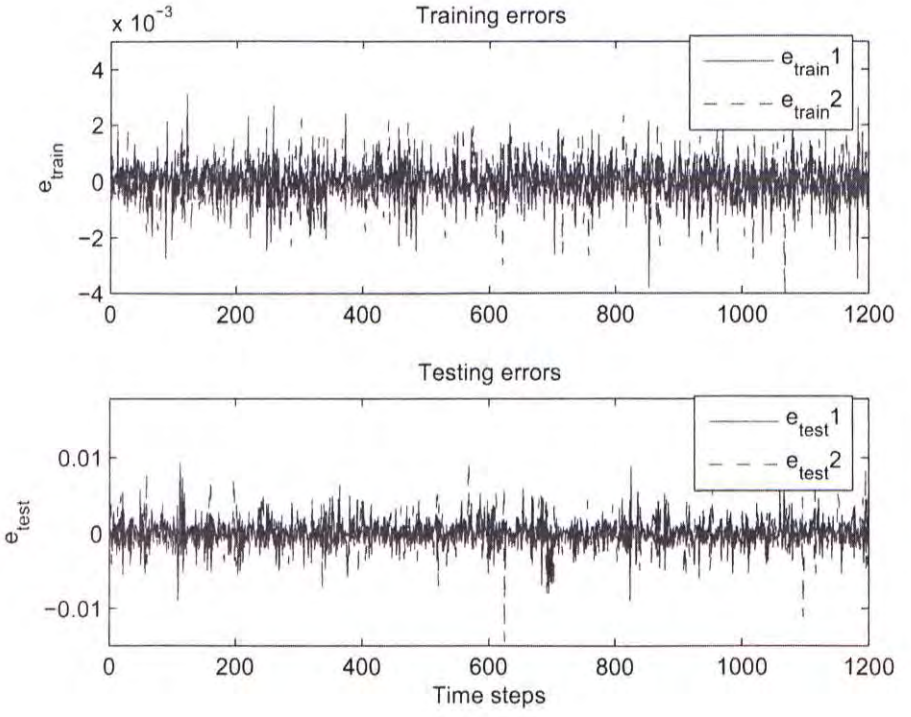

Figure 5.11: ESN training and testing errors in Example 3

□ **End of chapter.**

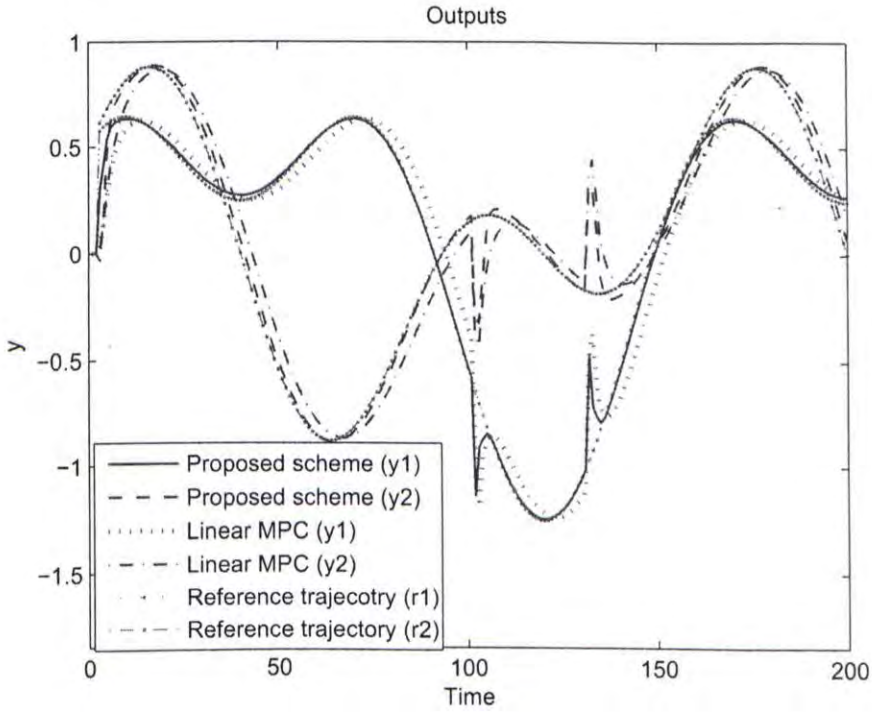

Figure 5.12: Outputs in Example 3

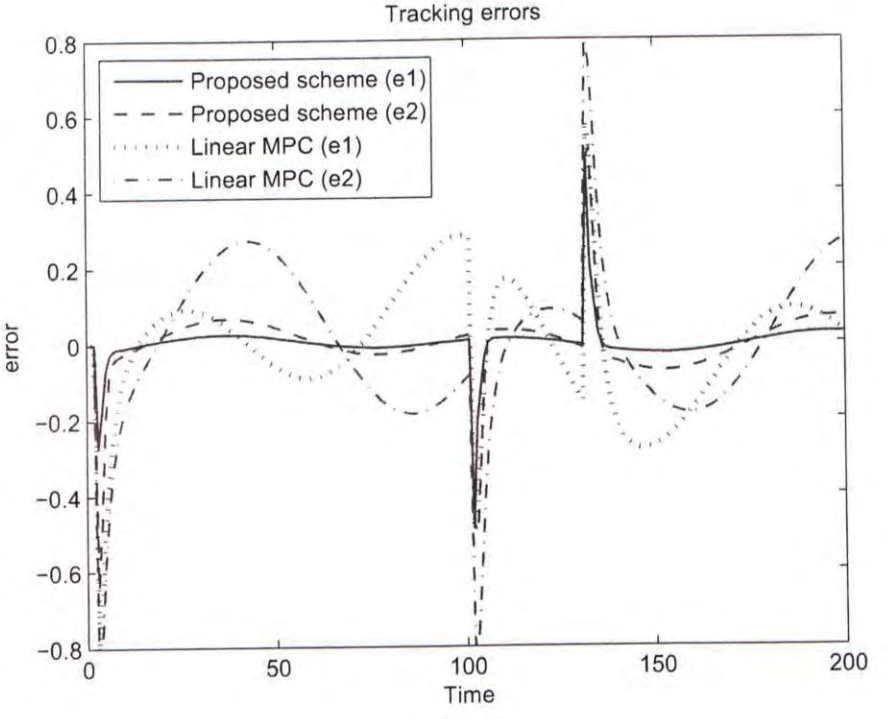

Figure 5.13: Tracking errors in Example 3

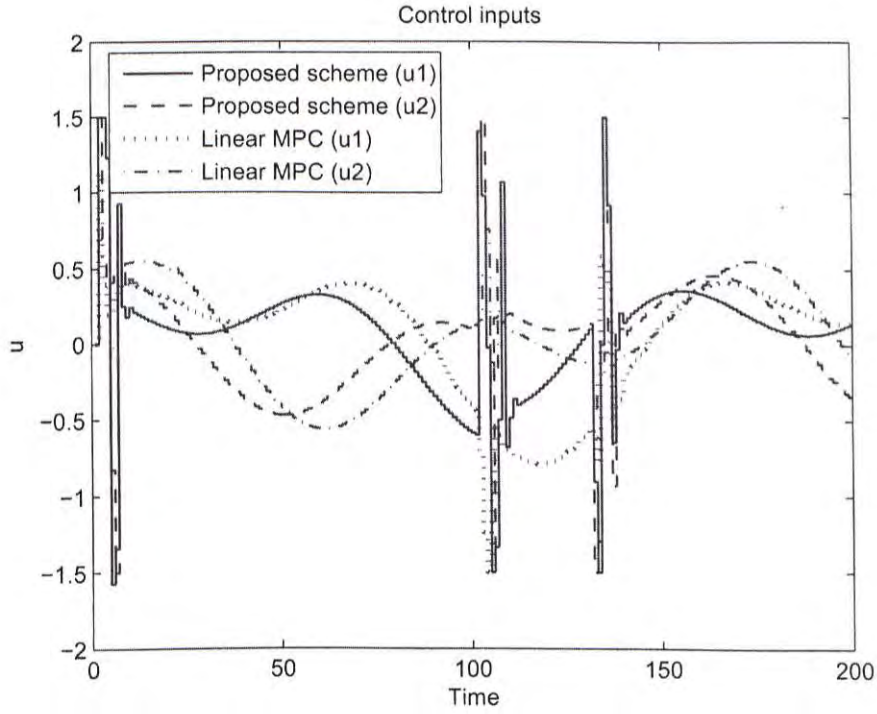

Figure 5.14: Control inputs in Example 3

Chapter 6

Model Predictive Control for Systems With Bounded Uncertainties Using a Discrete-Time Recurrent Neural Network

MPC that take consideration of uncertainties in the process model is called robust MPC. One way to deal with uncertainties in MPC is the worst case approach, which obtains a sequence of feedback control laws that minimizes the worst case cost. In industrial processes, it required the real-time solution to a minimax optimization problem. Although the robustness of MPC has been studied and is now well understood, the research outcomes are conceptual controllers that can work in principle but not suitable for hardware implementation [60]. As a result, further investigations on a more implementable controller are needed.

6.1 Problem Formulation

6.1.1 Process Model

Consider the following discrete-time linear system with global bounded uncertainties:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ y(k) &= Cx(k) + Dw(k), \end{aligned} \tag{6.1}$$

with the constraints

$$\begin{aligned} u_{\min} \leq & u(k) \leq u_{\max}, \\ \Delta u_{\min} \leq & \Delta u(k) \leq \Delta u_{\max}, \\ w_{\min} \leq & w(k) \leq w_{\max}, \\ y_{\min} \leq & y(k) \leq y_{\max}, \end{aligned} \tag{6.2}$$

where $k \geq 0$, $x(k) \in \Re^n$ is the state vector, $u(k) \in \Re^m$ is the input vector, and $y(k) \in \Re^p$ is the output vector. $w(k) \in \Re^q$ denotes the vector of bounded uncertainties. $u_{\min} \leq u_{\max}$, $w_{\min} \leq w_{\max}$, $y_{\min} \leq y_{\max}$ are vectors of upper and lower bounds.

6.1.2 Robust MPC Design

MPC is a step-by-step optimization technique: at each sampling time k, measure of estimate the current state, obtain the optimal input vector by solving a optimization problem. When bounded uncertainties are considered explicitly, a robust MPC law can be derived by minimizing the maximum cost within the model described by the uncertainty set. The optimal control action is obtained by solving a minimax optimization problem:

$$\min_{\Delta u} \max_{w} \quad J(\Delta u, w), \tag{6.3}$$

subjected to the constraints in (6.2).

The objective function $J(\Delta u, w)$ can be with an infinite or finite, linear or quadratic norm criterion. In this paper, we consider an objective function with a finite horizon quadratic criterion:

$$\begin{aligned} J(\Delta u, w) = & \sum_{j=1}^{N} [r(k+j|k) - y(k+j|k)]^T \Phi [r(k+j|k) - y(k+j|k)] + \\ & \sum_{j=0}^{N_u-1} [\Delta u(k+j|k)]^T \Psi [\Delta u(k+j|k)] \end{aligned} \tag{6.4}$$

where k is the current time step, $y(k+j|k)$ denotes the predicted output, $r(k+j|k)$ denotes the reference trajectory of output signal (desired output), and $\Delta u(k+j|k)$ denotes the input increment, where $\Delta u(k+j|k) = u(k+j|k) - u(k-1+j|k)$. $\Phi \in \Re^{p\times p}$, $\Psi \in \Re^{m\times m}$ are appropriate weighting matrices. N denotes the predictive horizon ($1 \leq N$). N_u denotes the control horizon ($0 < N_u \leq N$). After N_u control moves, $\Delta u(k+j|k)$ becomes zero.

According to the process model (6.1):

$$y(k+j) = CA^j x(k) + C\sum_{i=0}^{j-1} A^i Bu(k+j-i-1) + Dw(k+j), \quad j = 1, ..., N \tag{6.5}$$

Define following vectors:

$$\begin{aligned} \bar{y}(k) &= [y(k+1|k) \quad \cdots \quad y(k+N|k)]^T \in \Re^{Np}, \\ \bar{u}(k) &= [u(k|k) \quad \cdots \quad u(k+N_u-1|k)]^T \in \Re^{N_u m}, \\ \Delta\bar{u}(k) &= [\Delta u(k|k) \quad \cdots \quad \Delta u(k+N_u-1|k)]^T \in \Re^{N_u m}, \\ \bar{r}(k) &= [r(k+1|k) \quad \cdots \quad r(k+N|k)]^T \in \Re^{Np}, \end{aligned} \tag{6.6}$$

where the reference trajectory $\bar{r}(k)$ is known in advance. The predicted output $\bar{y}(k)$ is expressed in the following form:

$$\begin{aligned} \bar{y}(k) &= Sx(k) + M\bar{u}(k) + Ew(k) \\ &= Sx(k) + M\Delta\bar{u}(k) + Vu(k-1) + Ew(k), \end{aligned} \tag{6.7}$$

where

$$S = [CA \quad CA^2 \quad \cdots \quad CA^N]^T \in \Re^{Np\times n},$$

$$E = [D \quad D \quad \cdots \quad D]^T \in \Re^{Np \times q},$$

$$V = \begin{bmatrix} CB \\ C(A+I)B \\ \vdots \\ C(A^{N_u-1} + \cdots + A + I)B \\ C(A^{N_u} + \cdots + A + I)B \\ \vdots \\ C(A^{N-1} + \cdots + A + I)B \end{bmatrix} \in \Re^{Np \times m},$$

$$M = \begin{bmatrix} CB & \cdots & 0 \\ C(A+I)B & \cdots & 0 \\ \vdots & \ddots & \vdots \\ C(A^{N_u-1} + \cdots I)B & \cdots & CB \\ C(A^{N_u} + \cdots I)B & \cdots & C(A+I)B \\ \vdots & \ddots & \vdots \\ C(A^{N-1} + \cdots I)B & \cdots & C(A^{N-N_u} + \cdots I)B \end{bmatrix} \in \Re^{Np \times N_u m},$$

I denotes the identity matrix. Define vectors:

$$\Delta\bar{u}_{\min} = [\Delta u_{\min} \cdots \Delta u_{\min}]^T \in \Re^{N_u m}, \Delta\bar{u}_{\max} = [\Delta u_{\max} \cdots \Delta u_{\max}]^T \in \Re^{N_u m}$$

$$\bar{u}_{\min} = [u_{\min} \cdots u_{\min}]^T \in \Re^{N_u m}, \bar{u}_{\max} = [u_{\max} \cdots u_{\max}]^T \in \Re^{N_u m},$$

$$\bar{y}_{\min} = [y_{\min} \cdots y_{\min}]^T \in \Re^{N_u p}, \bar{y}_{\max} = [y_{\max} \cdots y_{\max}]^T \in \Re^{N_u p},$$

$$\tilde{I} = \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & I & I \end{bmatrix} \in \Re^{N_u m \times N_u m}.$$

Thus, the original minimax optimization can be expressed in the following

form:

$$\begin{aligned}
\min_{\Delta u}\max_{w} \quad & [Sx(k) - M\Delta\bar{u}(k) - Vu(k-1) - Ew(k)]^T\Phi[Sx(k) \\
& - M\Delta\bar{u}(k) - Vu(k-1) - Ew(k)] + \Delta\bar{u}^T(k)\Psi\Delta\bar{u}(k) \\
\text{s.t.} \quad & \bar{u}_{\min} \le u(k) + \tilde{I}\Delta\bar{u}(k) \le \bar{u}_{\max} \\
& \Delta\bar{u}_{\min} \le \Delta\bar{u}(k) \le \Delta\bar{u}_{\max} \\
& \bar{w}_{\min} \le \bar{w}(k) \le \bar{w}_{\max} \\
& \bar{y}_{\min} \le \bar{y}(k) + M(k)\Delta\bar{u}(k) \le \bar{y}_{\max}
\end{aligned} \tag{6.8}$$

By defining the variable vectors $u = \Delta\bar{u}(k) \in \Re^{N_u m}$, $w = w(k) \in \Re^q$. By neglecting the constraints on $u(k)$ and $y(k)$, the problem (6.8) can be rewritten as a minimax quadratic programming problem:

$$\begin{aligned}
\min_{u}\max_{w} \quad & \frac{1}{2}u^TQu + c^Tu - u^THw - \frac{1}{2}w^TRw - b^Tw \\
\text{s.t.} \quad & u \in \mathcal{U}, \quad w \in \mathcal{W}
\end{aligned} \tag{6.9}$$

where $\mathcal{U}$ and $\mathcal{W}$ are two box set defined as $\mathcal{U} = \{u \in \Re^{N_u m} | \Delta\bar{u}_{\min} \le u \le \Delta\bar{u}_{\max}\}$, $\mathcal{W} = \{w \in \Re^q | \bar{w}_{\min} \le w \le \bar{w}_{\max}\}$. The coefficient matrices and vectors are

$$Q = 2(M^T\Phi M + \Psi) \in \Re^{N_u m \times N_u m}, c = -2M^T\Phi(\bar{r}(k) - Sx(k) - Vu(k-1)) \in \Re^{N_u m},$$

$$R = 2E^T\Phi E \in \Re^{q\times q}, b = \Phi(\bar{r}(k) - Sx(k) - Vu(k-1)) \in \Re^q,$$

$$H = 2M^T\Phi E \in \Re^{N_u m \times q}$$

.

The solution to the minimax quadratic programming problem (9) gives the vector of control action $\Delta\bar{u}(k)$. The control law is given by $\bar{u}(k) = f(\Delta\bar{u}(k) + \bar{u}(k-1))$, where $f(\cdot)$ is defined as

$$f(\varepsilon_i) = \begin{cases} \varepsilon_i, & (G\varepsilon)_i \le l_i, \\ l_i, & (G\varepsilon)_i > l_i. \end{cases} \tag{6.10}$$

and G and l are defined as

$$G = [-\tilde{I} \quad \tilde{I} \quad -M \quad M]^T \in \Re^{(2N_u m + 2Np)\times N_u m},$$

$$l = \begin{bmatrix} -\bar{u}_{\min} + \bar{u}(k) \\ \bar{u}_{\max} - \bar{u}(k) \\ -\bar{y}_{\min} + \bar{y}(k) \\ \bar{y}_{\max} - \bar{y}(k) \end{bmatrix} \in \Re^{2N_u m + 2Np}.$$

The first element $u(k|k)$ is used as the control signal.

In industrial control processes, to solve large-scale minimax optimization problems in real-time is a major obstacle for robust MPC. In the next section, we will propose a recurrent neural network for solving (6.9).

6.2 Recurrent Neural Network Approach

6.2.1 Neural Network Model

Continuous-time neural networks for solving minimax problems has been investigated in [61]-[63]. However, in view of the availability of the digital hardware and the compatibility to the digital computers, discrete-time neural network is more desirable in practical implementation. In this section, we proposed a discrete-time recurrent neural network for minimax problem (6.9).

By the saddle point condition [64], (6.9) can be formulated as a linear variational inequality (LVI):

$$(s - s^*)^T (Ms^* + q) \geq 0, \quad \forall s \in \Omega, \tag{6.11}$$

where

$$M = \begin{bmatrix} Q & -H \\ H^T & R \end{bmatrix}, \quad q = \begin{bmatrix} c \\ b \end{bmatrix}, \quad \Omega = \mathcal{U} \times \mathcal{W}. \tag{6.12}$$

According to the well-known saddle point theorem [?], $s^* = (u^*, w^*)$ is a saddle point of $J(u, w)$ if satisfying

$$J(u^*, w) \leq J(u^*, w^*) \leq J(u, w^*), \quad \forall (u, w) \in \Omega. \tag{6.13}$$

We define the saddle point set $\Omega^* = \{(u^*, w^*) \in \Omega | (u^*, w^*) \text{ satisfy } (3.13)\}$ and assume Ω^* is not empty. It is obvious that if $(u^*, w^*) \in \Omega^*$, then (u^*, w^*) is the optimal solution to the minimax problem (3.9).

According to inequalities (6.13), we can get that v^* is a global minimizer of the objective function $J(v, w^*)$ with respect to $\mathcal{U}$, while w^* is the global minimizer of $J(v^*, w)$ with respect to $\mathcal{W}$. As a result, the following LVIs hold:

$$(u - u^*)^T(Qu^* + c - Hw^*) \geq 0, \quad \forall u \in \mathcal{U}, \tag{6.14}$$

$$(w - w^*)^T(Rw^* + b + H^T u^*) \geq 0, \quad \forall w \in \mathcal{W}. \tag{6.15}$$

According to the basic property of the projection mapping on a closed convex set:

$$[z - P_\Omega(z)]^T[P_\Omega(z) - v] \geq 0, \quad \forall z \in \Re, v \in \Omega. \tag{6.16}$$

Based on (6.14)-(6.16) and lemma 1 in [65], we can get that $(u^*, w^*) \in \Omega^*$ if and only if the following equations hold:

$$u^* = P_{\mathcal{U}}[u^* - \alpha(Qu^* + c - Hw^*)] \tag{6.17}$$

$$w^* = P_{\mathcal{W}}[w^* - \alpha(Rw^* + b + H^T u^*)] \tag{6.18}$$

where $\alpha > 0$ is a scaling constant, $P_{\mathcal{U}}(\cdot)$ and $P_{\mathcal{W}}(\cdot)$ are piecewise activation functions defined as:

$$P_{\mathcal{U}}(\varepsilon_i) = \begin{cases} \Delta u_{\min}, \varepsilon_i < \Delta u_{\min}; \\ \varepsilon_i, \quad \Delta u_{\min} \leq \varepsilon_i \leq \Delta u_{\max}; \\ \Delta u_{\max}, \varepsilon_i > \Delta u_{\max}. \end{cases} \quad P_{\mathcal{W}}(\varepsilon_i) = \begin{cases} w_{\min}, & \varepsilon_i < w_{\min}; \\ \varepsilon_i, & w_{\min} \leq \varepsilon_i \leq w_{\max}; \\ w_{\max}, & \varepsilon_i > w_{\max}. \end{cases} \tag{6.19}$$

Based on the equations (6.17) and (6.18), we propose a recurrent neural network for solving (6.9) as follow:

$$\begin{cases} u(t+1) = P_{\mathcal{U}}[u(t) - \alpha(Qu(t) + c - Hw(t))] \\ w(t+1) = P_{\mathcal{W}}[w(t) - \alpha(Rw(t) + b + H^T u(t))] \end{cases} \tag{6.20}$$

The proposed recurrent neural network has a simple structure, and can be easily implemented using digital hardware. In the next section, we will prove that the proposed neural network has global exponential convergence property under some mild conditions.

6.2.2 Convergence Analysis

Definition: Neural network (6.20) is said to be globally exponentially convergent to the equilibrium point (u^e, w^e) if both u^e and w^e satisfy

$$\begin{aligned} \|u(t) - u^e\| &\leq c_0\|u(0) - u^e\|e^{-\eta t}, \quad \forall t \geq 1; \\ \|w(t) - w^e\| &\leq b_0\|w(0) - w^e\|e^{-\eta t}, \quad \forall t \geq 1; \end{aligned} \tag{6.21}$$

where η is a positive constant independent of the initial point, c_0 and b_0 are positive constant dependent on the initial point.

Lemma 1: The neural network (6.20) has a unique equilibrium point, which is the saddle point of $J(u, w)$.

Proof: Similar to the proof in [65], we can establish that the neural network (6.20) has a unique equilibrium point (u^e, w^e).

Define a equilibrium point set $\Omega^e = \{(u^e, w^e) \in \Omega | (u^e, w^e) \text{ satisfy } (6.17) and (6.18))\}$. According to the above derivation, it is obvious that the equations (6.17) and (6.18) is equivalent to (6.13) for all $(u, w) \in \Omega$, from the definition of Ω^*, we can get that $\Omega^e = \Omega^*$, which means the equilibrium point of (6.20) is the saddle point of $J(u, w)$.

Lemma 2: For all $z \in \Re^n$,

$\|P_{\mathcal{U}}(v) - P_{\mathcal{U}}(z)\|^2 \leq \|v - z\|^2, \quad \|P_{\mathcal{W}}(v) - P_{\mathcal{W}}(z)\|^2 \leq \|v - z\|^2.$

Proof: From the inequality (6.16) we can easily prove that

$$\begin{aligned} &\|P_{\mathcal{U}}(v) - P_{\mathcal{U}}(z)\|^2 \leq (v - z)^T[P_{\mathcal{U}}(v) - P_{\mathcal{U}}(z)] \leq \|v - z\|^2, \\ &\|P_{\mathcal{W}}(v) - P_{\mathcal{W}}(z)\|^2 \leq (v - z)^T[P_{\mathcal{W}}(v) - P_{\mathcal{W}}(z)] \leq \|v - z\|^2, \quad \forall v, z \in \Re^n. \end{aligned} \tag{6.22}$$

Define $\lambda_i^Q > 0 (i = 1, ..., N_u m)$, $\lambda_j^R > 0 (j = 1, ..., Nq)$ as the eigenvalues of Q, R respectively, let $\lambda_{\min}^Q, \lambda_{\max}^Q, \lambda_{\min}^R, \lambda_{\max}^R$ be the smallest and largest eigenvalues of Q and R. Define two functions

$$\psi^Q(\alpha) = \begin{cases} 1 - \lambda_{\min}^Q \alpha, & 0 < \alpha \leq 2/(\lambda_{\min}^Q + \lambda_{\max}^Q) \\ \lambda_{\max}^Q \alpha - 1, & 2/(\lambda_{\min}^Q + \lambda_{\max}^Q) \leq \alpha < +\infty \end{cases} \tag{6.23}$$

$$\psi^R(\alpha) = \begin{cases} 1 - \lambda_{\min}^R \alpha, & 0 < \alpha \leq 2/(\lambda_{\min}^R + \lambda_{\max}^R) \\ \lambda_{\max}^R \alpha - 1, & 2/(\lambda_{\min}^R + \lambda_{\max}^R) \leq \alpha < +\infty \end{cases} \tag{6.24}$$

Then we give the following lemma:

Lemma 3: $\psi^Q(\alpha) < 1$ and $\psi^R(\alpha) < 1$ if and only if

$$0 < \alpha < \min\{2/\lambda_{\max}^Q, 2/\lambda_{\max}^R\}. \tag{6.25}$$

Proof: From the Theorem 2 in [66], we can get that $\psi^Q(\alpha) < 1$ if and only if $\alpha \in (0, 2/\lambda_{\max}^Q)$, similarly, $\psi^R(\alpha) < 1$ if and only if $\alpha \in (0, 2/\lambda_{\max}^R)$. We can easily verify that the sufficient and necessary condition for both $\psi^Q(\alpha) < 1$ and $\psi^R(\alpha) < 1$ is $0 < \alpha < \min\{2/\lambda_{\max}^Q, 2/\lambda_{\max}^R\}$.

Theorem 1: With any α that satisfies (6.25), the neural network (6.20) is globally exponentially convergent to the saddle point of $J(u, w)$.

Proof: From (6.23) and (6.24), we can obtain that $\psi^Q(\alpha) = \max\{(1 - \alpha\lambda_1^Q)^2, ..., (1 - \alpha\lambda_{N_u m}^Q)^2\}$, $\psi^R(\alpha) = \max\{(1 - \alpha\lambda_1^R)^2, ..., (1 - \alpha\lambda_{Nq}^R)^2\}$.

By Lemma 2:

$$
\begin{aligned}
\|u(k)-u^*\|^2 =& \|P_{\mathcal{U}}[u(t-1)-\alpha(Qu(t-1)+c-Hw(t-1))]- \\
& P_{\mathcal{U}}[u^*-\alpha(Qu^*+c-Hw^*)]\|^2 \\
\leq& \|(I-\alpha Q)(u(t-1)-u^*)\|^2 \\
\leq& \max\{(1-\alpha\lambda_1^Q)^2,...,(1-\alpha\lambda_{N_u m}^Q)^2\}\|u(t-1)-u^*\|^2 \\
=& \psi^Q(\alpha)^2\|u(t-1)-u^*\|^2 \\
\Longrightarrow \|u(t)-u^*\| \leq& \psi^Q(\alpha)\|u(t-1)-u^*\| \\
\leq& \psi^Q(\alpha)^t\|u(0)-u^*\| \\
\leq& e^{-\eta^Q(\alpha)t}\|u(0)-u^*\|
\end{aligned}
\tag{6.26}
$$

Similarly, $\|w(t)-w^*\| \leq e^{-\eta^R(\alpha)t}\|w(0)-w^*\|$. From Lemma 3, $\eta^Q(\alpha) > 0$ ($\psi^Q(\alpha) < 1$) and $\eta^R(\alpha) > 0$ ($\psi^R(\alpha) < 1$) for all α that satisfy (6.25).

From the above proof and lemma 1, we can obtain that for any α that satisfies (6.25), the neural network (6.20) is globally exponentially convergent to the unique equilibrium point (u^*, w^*), which is the saddle point of $J(u, w)$.

6.2.3 Control Scheme

The control scheme based on proposed recurrent neural network can be summarized as follows:

1. Let $k = 1$. Set terminal time T, sample time t, predictive horizon N, control horizon N_u, weighting matrices Φ and Ψ.

2. Calculate process model matrices S, E, V, M, neural network parameters Q, R, H, c, b.

3. Solve the quadratic minimax problems (3.9) using the proposed recurrent neural network, obtaining the optimal control action $\Delta\bar{u}(k)$.

4. Calculate the optimal input vector $\bar{u}(k) = f(\Delta\bar{u}(k) + \bar{u}(k-1))$, the first element $u(k|k)$ is sent to the process.

5. If $k < T$, set $k = k + 1$, return to step 2; otherwise, end.

6.3 Simulation Results

Consider a two-tank system described in [67], which is a two-input, two-output system, with the flow rates of the two inlet streams as the two inputs, and the liquid level in each tank as the two output variables.

By sampling at 0.2 min using a zero-order holder, the following discrete-time state-space model can be obtained:

$$\begin{aligned} x(k+1) &= \begin{bmatrix} -\frac{0.5}{3} & \frac{0.2}{3} \\ \frac{0.5}{2} & -\frac{0.5}{2} \end{bmatrix} x(k) + \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 2 \\ 1 \end{bmatrix} w(k) \end{aligned} \tag{6.27}$$

The set-point for the liquid levels (output) of tanks 1 and 2 are 0.8 and 0.7, respectively; the prediction and control horizons are $N = 10$ and $Nu = 4$; weighting matrices $\Phi = I$, $\Psi = 5I$; scaling constant $\alpha = 0.2$; an uncertainty $-0.02 \leq w \leq 0.02$ is considered to affect both liquid levels of tanks 1 and 2; moreover, the following constraints are considered:

$$\begin{gathered} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq u(k) \leq \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq y(k) \leq \begin{bmatrix} 0.6 \\ 0.7 \end{bmatrix} \\ \begin{bmatrix} -0.05 \\ -0.05 \end{bmatrix} \leq \Delta u(k) \leq \begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix} \\ -0.02 \leq w \leq 0.02 \end{gathered} \tag{6.28}$$

In order to compare the effectiveness and efficiency of the proposed approach, a linear matrix inequalities (LMI) approach [5] is also applied to the process. The simulation results are showed in Figs. 6.1 - 6.4. We can see that the proposed

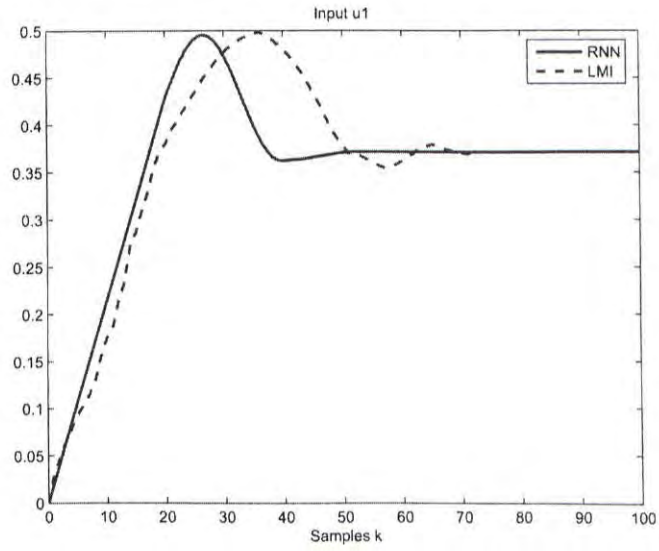

Figure 6.1: Input signals of tanks 1 using the proposed RNN approach and LMI approach

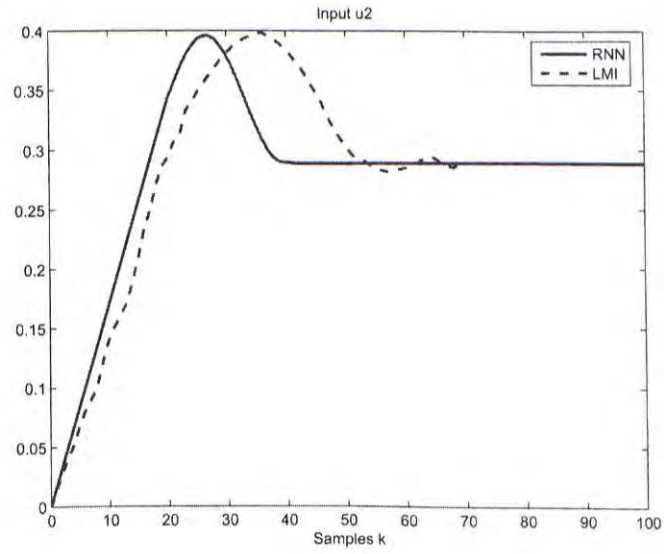

Figure 6.2: Input signals of tanks 2 using the proposed RNN approach and LMI approach

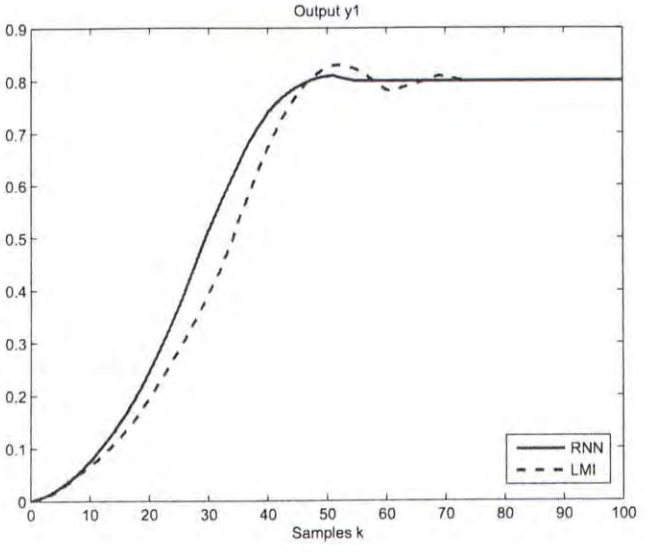

Figure 6.3: Output responses of tanks 1 using the proposed RNN approach and LMI approach

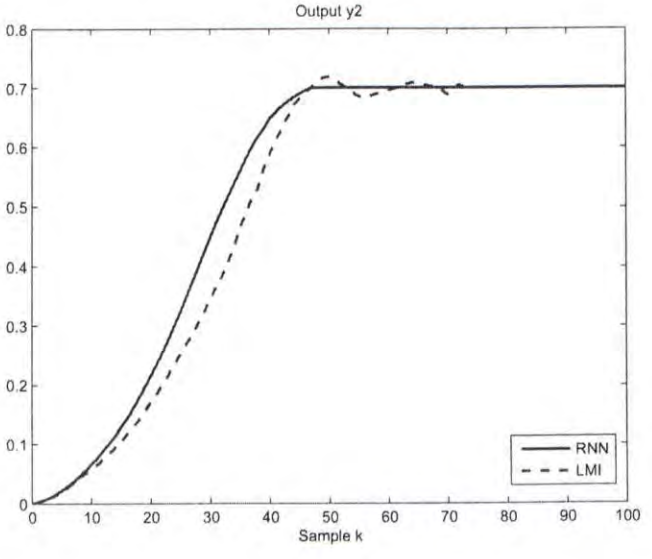

Figure 6.4: Output responses of tanks 2 using the proposed RNN approach and LMI approach

neural network approach gives a better set-point tracking performance with faster stable output responses.

□ **End of chapter.**

Chapter 7

Conclusions and Future Works

The preceding chapters addressed the synthesis, analysis, and applications of model predictive control (MPC) based on recurrent neural networks (RNNs). In this concluding chapter, we will summarize what has been accomplished in this research and describe some potential future works to extend the present results for the discussed problems.

7.1 Concluding Remarks

In this thesis, we have introduced the RNN approaches to MPC for linear systems, nonlinear systems, and systems with uncertainties.

The desirable features of RNN, such as global convergence and low complexity, have been utilized for MPC design. In Chapter 2 and 3, RNNs have been applied for solving the quadratic programming (QP) or linear programming (LP) problems associated with MPC at each sample time. Compared with existing related works [43][44], the proposed RNN-based approaches have simpler structures and do not suffer from local minima. Simulation results have shown that the RNN-based MPC schemes are effective and efficient.

More challenging works have been done for nonlinear MPC using RNNs in Chapter 4 and 5. By means of decomposition, the original optimization problem

associated with nonlinear MPC has been reformulated as a QP problem with a unknown nonlinear term. A recursive algorithm is developed for RNN learning, which guarantees the convergence of the solution to the original optimization problem. For unknown dynamic systems, the echo state network (ESN) has been used for system identification. The proposed RNN-based nonlinear MPC schemes have shown better performances than linear MPC in reference tracking and disturbance rejection.

In Chapter 6, the MPC synthesis problem for linear systems with bounded uncertainties has been formulated as a quadratic minimax problem, we have developed a discrete-time RNN for minimax optimization and proved its exponential convergence property. Compared with the linear matrix inequalities (LMI) approach, the proposed RNN-based robust MPC has shown superior performance.

Due to RNN's desirable features, the proposed schemes are efficient and suitable for real-time MPC implementation in industrial applications. The RNN-based MPC operates in a massively parallel fashion, which can be applied to large-scale multi-variable systems.

7.2 Future works

There are also many unsolved problems related to the analysis and synthesis of RNN-based MPC schemes. For future researches, the following possible works require further investigations.

1. For nonlinear systems with uncertainties, it is highly desirable to design robust nonlinear MPC systems based on RNNs to withstand possible parameter perturbation and random input disturbance.

2. As time delays may introduce detrimental effects in dynamic systems. To reduce or remove the effects, it is necessary to develop the design procedures for RNN-based MPC systems with time delays based on the results

of neurodynamic analysis.

3. In terms of applications, the RNN-based MPC schemes may be applied to large-scale multivariable systems, further applications include industrial process control, spacecraft attitude control, etc.

□ **End of chapter.**

Bibliography

[1] J. Richalet, A. Testud, L. J., and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, pp. 413-428, 1978.

[2] S. J. Qin and Thomas A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, pp. 733-764, 2003.

[3] E. F. Camacho and C. Bordons, *Model Predictive Control*, Springer, London, U.K., 2004.

[4] D. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, New York: Academic, 1982.

[5] M. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear Programming: Theory and Algorithms (2nd Ed.)*, New York: John Wiley, 1993.

[6] J. Rosen, "The gradient projection method for nonlinear programming. part i. linear constraints," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 1, pp. 181-217, 1960.

[7] W. Zangwill, "Non-linear programming via penalty functions," *Management Science*, vol. 13, no. 5, pp. 344-358, 1967.

[8] M. Rybashov, "The gradient method of solving convex programming problems on electronic analog computers," *Automation Remote Control*, vol. 26, no. 11, pp. 1886-1898, 1965.

[9] J. Hopfield, "Neural networks and physical systems with emergent collective computational properties," *Proc. National Academy of Science*, USA, Biophysics, vol. 79, pp. 2554-2588, 1982.

[10] J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. National Academy of Science*, USA, vol. 81, no. 10, pp. 3088-3092, 1984.

[11] J. Hopfield and D. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, no. 3, pp. 141-152, 1985.

[12] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, London, U.K. Wiley, 1993.

[13] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst*, vol. CAS-33, pp. 533-541, May 1986.

[14] M. Kennedy and L. Chua, "Neural networks for nonlinear programming," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 5, pp. 554-562, 1988.

[15] S. Zhang and A. Constantinides, "Lagrange programming neural networks," *IEEE Transactions on Circuits and Systems-II*, vol. 39, no. 7, pp. 441-452, 1992.

[16] Y. Xia, "Global convergence analysis of lagrangian networks," *IEEE Transactions on Circuits and Systems-I*, vol. 50, no. 6, pp. 818-822, 2003.

[17] J. Wang, "Analysis and design of a recurrent neural network for linear programming," *IEEE Transactions on Circuits and Systems-I*, vol. 40, no. 9, pp. 613-618, 1993.

[18] J. Wang, "A deterministic annealing neural network for convex programming," *Neural Networks*, vol. 7, no. 4, pp. 629-641, 1994.

[19] Y. Xia and J. Wang, "Primal neural networks for solving convex quadratic programs," *International Joint Conference on Neural Networks 1999*, vol. 1, pp. 582-587, 1999.

[20] Y. Xia, "A new neural network for solving linear and quadratic programming problems," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1544-1548, 1996.

[21] Y. Xia, "A new neural network for solving linear programming and quadratic programming problems," *Neural Networks*, vol. 9, no. 6, pp. 1544-1547, 1996.

[22] J. Wang, "Primal and dual assignment networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 784-790, 1997.

[23] J. Wang and Y. Xia, "Analysis and design of primal-dual assignment networks," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 183-194, 1998.

[24] Y. Xia, G. Feng, and J. Wang, "A primal-dual neural network for online resolving constrained kinematic redundancy in robot motion control," *IEEE Transactions on Systems, Man and Cybernetics-B*, vol. 35, no. 1, pp. 54-64, 2005.

[25] Y. Xia and J. Wang, "A dual neural network for kinematic control of redundant robot manipulators," *IEEE Transactions on Systems, Man and Cybernetics-B*, vol. 31, no. 1, pp. 147-154, 2001.

[26] Y. Zhang and J. Wang, "A dual neural network for convex quadratic programming subject to linear equality and inequality constraints," *Physics Letters A*, vol. 298, no. 4, pp. 271-278, 2002.

[27] Y. Zhang, J. Wang, and Y. Xia, "A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 658-667, 2003.

[28] S. Liu and J. Wang, "A simplified dual neural network for quadratic programming with its KWTA application," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1500-1510, Nov. 2006.

[29] T. Friesz, D. Bernstein, N. Mehta, R. Tobin, and S. Ganjalizadeh, "Day-to-day dynamic network disequilibria and idealized traveler information systems," *Operations Research*, vol. 42, no. 6, pp. 1120-1136, 1994.

[30] Y. Xia and J. Wang, "On the stability of globally projected dynamical systems," *Journal of Optimization Theory and Applications*, vol. 106, no. 1, pp. 129-150, 2000.

[31] Y. Xia and J. Wang, "Global asymptotic and exponential stability of a dynamic neural system with asymmetric connection weights," *IEEE Transactions on Automatic Control*, vol. 46, no. 4, pp. 635-638, 2001.

[32] Y. Xia, H. Leung, and J. Wang, "A projection neural network and its application to constrained optimization problems," *IEEE Transactions Circuits and Systems-I*, vol. 49, no. 4, pp. 447-458, 2002.

[33] Y. Xia and J. Wang, "A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints," *IEEE Transactions on Circuits and Systems I*, vol. 51, no. 7, pp. 1385-1394, 2004.

[34] Y. Xia, G. Feng, and J. Wang, "A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equations," *Neural Networks*, vol. 17, no. 7, pp. 1003-1015, 2004.

[35] Y. Xia, "Further results on global convergence and stability of globally projected dynamical systems," *Journal of Optimization Theory and Applications*, vol. 122, no. 3, pp. 627-649, 2004.

[36] Y. Xia, "An extended projection neural network for constrained optimization," *Neural Computation*, vol. 16, pp. 863-883, 2004.

[37] M. Forti, P. Nistri, and M. Quincampoix, "Generalized neural network for nonsmooth nonlinear programming problems," *IEEE Transactions on Circuits and Systems-I*, vol. 51, no. 9, pp. 1741-1754, 2004.

[38] Q. Liu, J. Cao, and Y. Xia, "A delayed neural network for solving linear projection equations and its analysis," *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 834-843, 2005.

[39] Y. Yang and J. Cao, "Solving quadratic programming problems by delayed projection neural network," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1630-1634, 2006.

[40] Q. Liu, J. Wang, and J. Cao, "A delayed lagrangian network for solv- ing quadratic programming problems with equality constraints," *Lecture Notes In Computer Science*, vol. 3971, pp. 369-378, Springer, 2006. ISNN2006.

[41] Q. Liu and J. Wang, "A one-layer recurrent neural network with a discontinuous activation function for linear programming," *Neural Computation*, vol. 20, no. 5, pp. 1366-1383, 2008.

[42] Q. Liu and J. Wang, "A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming," *IEEE Transactions on Neural Networks*, vol. 19, no. 4, pp. 558-570, 2008.

[43] J. M. Quero and E. F. Camacho, "Neural network for constrained predictive control," *IEEE Trans. Circuits Syst. I.*, vol. 40, no. 9, pp. 621-626, May, 1993.

[44] L. Wang and F. Wan, "Structured neural networks for constrained model predictive control," *Automatica*, vol. 37, pp. 1235-1243, 2001.

[45] L. Cheng, Z. Hou, and M. Tan, "Constrained multi-variable generalized predictive control using a dual neural network," *Neural Comput. & Applic.*, vol. 16, pp. 505-512, 2007.

[46] Y. Pan and J.Wang, "Two neural network approaches to model predictive control," *Proceedings of the American Control Conference*, Seattle, Washington, USA, pp. 1685-1690, 2008.

[47] Y. Pan and J.Wang, "Robust model predictive control using a discrete-time recurrent neural network," *Advances in Neural Networks - ISNN2008*, Springer-Verlag, vol. 5263, pp. 883-892, 2008.

[48] C. Venkateswarlu and K. Rao, "Dynamic recurrent radial basis function network model predictive control of unstable nonlinear processes," *Chemical Engineering Science*, vol. 60, pp. 6718-6732, 2005.

[49] U. Yuzgec, Y. Becerikli, and M. Turker, "Dynamic neural-network-based model-predictive control of an industrial baker's yeast drying process," *IEEE Trans Neural Netw.*, vol. 19, pp. 1231-1242, 2008.

[50] Y. Pan and J.Wang, "Nonlinear model predictive control using a recurrent neural network," *Proceedings of the 2008 International Joint Conference on Neural Networks*, Hong Kong, pp. 2297-2302, June 2008.

[51] S. Boyd and L. Vandenbeghe, *Convex Optimization*, Cambridge, U.K., Cambridge Univ. Press, 2004.

[52] L. A. Zadeh and L. H. Whalen, "On optimal control and linear programming," *IEEE Trans. Automat. Contr.*, vol. AC-7, pp. 45-46, Jan. 1962.

[53] A. I. Propoi, "Use of linear programming methods for synthesizing sampled-data automatic systems," *Automat. Rem. Control.*, vol. 24, no. 7, pp. 837-844, 1963.

[54] T. S. Chang and D. E. Seborg, "A linear programming approach for multivariable feedback control with inequality constraints," *Int. J. Control*, vol. 37, no. 3, pp. 583-597, 1983.

[55] H. Genceli and M. Nikolaou, "Robust stability analysis of constrained ' - norm model predictive control," *AIChE J.*, vol. 39, no. 12, pp. 1954-1965, 1993.

[56] C. V. Rao and J. B. Rawlings, "Linear programming and model predictive control", *J. Process Control*, vol. 10, pp. 283-289, 2000.

[57] A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming - the explicit solution," *IEEE Trans. Automatic Control*, vol. 47, pp. 1974-1985, 2002.

[58] P. J. Campo and M. Morari, "Model predictive optimal averaging level control," *AIChE J.*, vol. 35, no. 4, pp. 579-591, 1989.

[59] C. V. Rao and J. B. Rawlings, "Linear programming and model predictive control", *J. Process Control*, vol. 10, pp. 283-289, 2000.

[60] D. Mayne, J. Rawlings, C. Rao, P. Scokaert, "Constrained model predictive control: Stability and optimality". *Automatica*, vol. 36, pp. 789-814, 2000.

[61] Q. Tao and T. Fang, "The neural network model for solving minimax problems with constraints". *Control Theory Applicat.*, vol. 17, pp. 82-84, 2000.

[62] X. Gao and L. Liao, "A neural network for a class of convex quadratic minimax problems with constraints," *IEEE Trans. Neural Netw.*, vol. 16, pp. 622-628, 2004.

[63] X. Gao and L. Liao, "A novel neural network for a class of convex quadratic minimax problems," *Neural Computation*, vol. 18, pp. 1818-1846, 2006.

[64] M. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear programming: theory and algorithms*, New York: Wiley, 1993.

[65] M. Perez-Ilzarbe,, "Convergence analysis of a discrete-time recurrent neural network toperform quadratic real optimization with bound constraints," *IEEE Trans. Neural Netw.*, vol. 9, pp. 1344-1351, 1998.

[66] K. Tan, H. Tang and Z. Yi, "Global exponential stability of discrete-time neural networks for constrained quadratic optimization," *Neurocomputing*, vol. 56, pp. 399-506, 2004.

[67] T. Alamo, D. Ramirez and E. Camacho, "Efficient implementation of constrained min–max model predictive control with bounded uncertainties: a vertex rejection approach," *Journal of Process Control*, vol. 15, pp. 149-158, 2005.

[68] K. Hornik, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359-366, 1989.

[69] M. Leshno, V. Y. Lin, A. Pinkus, and S. schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Netw.*, vol. 6, no. 6, pp. 861-867, 1993.

[70] D. R. Hush and B. Horne, "Efficient algorithms for function approximation with piecewise linear sigmoid networks," *IEEE Trans. Neural Netw.*, vol. 9, no. 6, pp. 1129-1141, 1998.

[71] H. Li and H. Deng, "An approximate internal model-based neural control for unknown nonlinear discrete processes," *IEEE Trans. Neural Netw.*, vol. 17, pp. 659-670, 2006.

[72] H. Jaeger, "The 'echo' state approach to analyzing and training recurrent neural networks," *Technical report GMD*, report 148, 2001.

[73] H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication," *Science*, pp. 78-80, April 2, 2004.

[74] D. Xu, J. Lan and J. C. Principe, "Direct adaptive control: An echo state network and genetic algorithm approach," *Proceedings of the 2005 International Joint Conference on Neural Networks*, Montreal, pp. 1483-1486, 2005.

[75] M. D. Skowronski and J. G. Harris, "Automatic speech recognition using a predictive echo state network classifier," *Neural Networks*, vol. 20, pp. 414-423, 2007.

[76] F. J. Doyle III, B. A. Ogunnaike, and R. K. Pearson, "Nonlinear model-based control using second-order volterra models," *Automatica*, vol. 31, no. 5, pp. 697-714, 1995.

[77] F. Wu, "LMI-based robust model predictive control and its applications to an industrial CSTR problem," *Journal of Process Control*, vol. 11, pp. 649-659, 2001.

[78] D. Chwa, "Sliding-mode tracking control of nonholonomic wheeled mobile robots in polar coordinates," *IEEE Trans. Control Syst. Technol.*, vol. 12, no. 4, pp. 637-644, Apr. 2004.

[79] C. D. Sousa, E. M. Hemerly, and R. K. H. Galvao, "adaptive control for mobile robot using wavelet networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 32, no. 4, pp. 493-504, Apr. 2002.

[80] D. Gu and H. Hu, "Neural predictive control for a car-like mobile robot," *Robot. Autonom. Syst.*, vol. 39, no. 2, pp. 73-86, 2002.

[81] J. S. Oh, J. B. Park, and Y. H. Choi, "Stable path tracking control of a mobile robot using a wavelet-based fuzzy neural network," *Int. J. Contr., Autom. Syst.* , vol. 3, no. 4, pp. 552-563, 2005.

[82] S. Kim, J. Park, and J. Lee, "Implementation of tracking and capturing a moving object using a mobile robot," *Int. J. Contr., Autom. Syst.* , vol. 3, no. 3, pp. 444-452, 2005.

[83] J. S. Yoo, Y. H. Choi, and J. B. Park, "Generalized predictive control based on self-recurrent wavelet neural network for stable path tracking of mobile robots: adaptive learning rates approach," *IEEE Trans. Circ. Syst. I*, vol. 53, no. 6, pp. 1381-1394, June 2006.