2

Mathematical Preliminaries

In this chapter, important mathematical preliminaries, required in future chapters, are presented.

2.1 Stability Definitions

Consider an MIMO nonlinear system:

$$x(k+1) = F(x(k), u(k)) \tag{2.1}$$

$$y(k) = h(x(k)), \tag{2.2}$$

where $x \in \Re^n$, $u \in \Re^m$, and $F \in \Re^n \times \Re^m \to \Re^n$ is nonlinear function.

Definition 2.1. *The system* (2.1) *is said to be forced or to have input. In contrast, the system described by an equation without explicit presence of an input* u*, that is,*

$$x(k+1) = F(x(k))$$

is said to be unforced. It can be obtained after selecting the input u *as a feedback function of the state*

$$u(k) = \xi(x(k)). \tag{2.3}$$

Such substitution eliminates u*:*

$$x(k+1) = F(x(k), \xi(x(k))), \tag{2.4}$$

and yields an unforced system (2.4) [9].

Definition 2.2. *The solution of* (2.1)–(2.3) *is semiglobally uniformly ultimately bounded (SGUUB), if for any* Ω*, a compact subset of* $\Re^n$ *and all* $x(k_0) \in \Omega$*, there exists an* $\varepsilon > 0$ *and a number* $N(\varepsilon, x(k_0))$ *such that* $\|x(k)\| < \varepsilon$ *for all* $k \geq k_0 + N$.

E.N. Sanchez et al.: *Discrete-Time High Order Neural Control – Trained with Kalman Filtering*, Studies in Computational Intelligence (SCI) **112**, 5–10 (2008)
www.springerlink.com

In other words, the solution of (2.1) is said to be SGUUB if, for any a priori given (arbitrarily large) bounded set Ω and any a priori given (arbitrarily small) set Ω_0, which contains $(0,0)$ as an interior point, there exists a control (2.3) such that every trajectory of the closed loop system starting from Ω enters the set $\Omega_0 = \{x(k) | \|x(k)\| < \varepsilon\}$ in a finite time and remains in it thereafter.

Theorem 2.1. *Let $V(x(k))$ be a Lyapunov function for a discrete-time system (2.1), which satisfies the following properties:*

$$\begin{aligned} \gamma_1(\|x(k)\|) \leq V(x(k)) \leq \gamma_2(\|x(k)\|), \\ V(x(k+1)) - V(x(k)) = \Delta V(x(k)) \\ \leq -\gamma_3(\|x(k)\|) + \gamma_3(\zeta), \end{aligned}$$

where ζ is a positive constant, $\gamma_1(\bullet)$ and $\gamma_2(\bullet)$ are strictly increasing functions, and $\gamma_3(\bullet)$ is a continuous, nondecreasing function. Thus if

$$\Delta V(x) < 0 \quad for \quad \|x(k)\| > \zeta,$$

then $x(k)$ is uniformly ultimately bounded, i.e., there is a time instant k_{T}, such that $\|x(k)\| < \zeta, \forall k < k_{\mathrm{T}}$.

Definition 2.3. *A subset $S \in \Re^n$ is bounded if there exists $r > 0$ such that $\|x\| \leq r$ for all $x \in S$ [9].*

Theorem 2.2 (Separation Principle). [12]. *The asymptotic stabilization problem of the system (2.1)–(2.2), via estimated state feedback*

$$\begin{aligned} u(k) &= \xi(\widehat{x}(k)), \\ \widehat{x}(k+1) &= F(\widehat{x}(k), u(k), y(k)) \end{aligned} \tag{2.5}$$

is solvable if and only if the system (2.1)–(2.2) is asymptotically stabilizable and exponentially detectable.

Corollary 2.1. [12]. *There is an exponential observer for a Lyapunov stable discrete-time nonlinear system (2.1)–(2.2) with $u = 0$ if and only if the linear approximation*

$$\begin{aligned} x(k+1) &= A(k)x(k) + Bu(k), \\ y(k) &= Cx(k), \end{aligned} \tag{2.6}$$

$$A = \left.\frac{\partial F}{\partial x}\right|_{x=0}, \quad B = \left.\frac{\partial F}{\partial u}\right|_{x=0}, \quad C = \left.\frac{\partial h}{\partial x}\right|_{x=0}$$

of the system (2.1)–(2.2) is detectable.

2.2 Discrete-Time High Order Neural Networks

The use of multilayer neural networks is well known for pattern recognition and for modeling of static systems. The NN is trained to learn an input–output map. Theoretical works have proven that, even with just one hidden layer, a NN can uniformly approximate any continuous function over a compact domain, provided that the NN has a sufficient number of synaptic connections.

For control tasks, extensions of the first-order Hopfield model called recurrent high order neural networks (RHONN), which present more interactions among the neurons, are proposed in [13, 16]. Additionally, the RHONN model is very flexible and allows to incorporate to the neural model a priori information about the system structure.

Consider the following discrete-time RHONN:

$$\widehat{x}_i(k+1) = w_i^\top z_i(\widehat{x}(k), \upsilon(k)), \quad i = 1, \cdots, n, \tag{2.7}$$

where $\widehat{x}_i$ $(i = 1, 2, \cdots, n)$ is the state of the ith neuron, L_i is the respective number of high order connections, $\{I_1, I_2, \cdots, I_{L_i}\}$ is a collection of nonordered subsets of $\{1, 2, \cdots, n+m\}$, n is the state dimension, m is the number of external inputs, w_i $(i = 1, 2, \cdots, n)$ is the respective online adapted weight vector, and $z_i(\widehat{x}(k), \varrho(k))$ is given by

$$z_i(x(k), \varrho(k)) = \begin{bmatrix} z_{i_1} \\ z_{i_2} \\ \vdots \\ z_{i_{L_i}} \end{bmatrix} = \begin{bmatrix} \prod_{j \in I_1} \xi_{i_j}^{d_{ij}(1)} \\ \prod_{j \in I_2} \xi_{i_j}^{d_{ij}(2)} \\ \vdots \\ \prod_{j \in I_{L_i}} \xi_{i_j}^{d_{ij}(L_i)} \end{bmatrix}, \tag{2.8}$$

with $d_{j_i}(k)$ being a nonnegative integers, and ξ_i defined as follows:

$$\xi_i = \begin{bmatrix} \xi_{i_1} \\ \vdots \\ \xi_{i_1} \\ \xi_{i_{n+1}} \\ \vdots \\ \xi_{i_{n+m}} \end{bmatrix} = \begin{bmatrix} S(x_1) \\ \vdots \\ S(x_n) \\ \varrho_1 \\ \vdots \\ \varrho_m \end{bmatrix}. \tag{2.9}$$

In (2.9), $\varrho = [\varrho_1, \varrho_2, \ldots, \varrho_m]^\top$ is the input vector to the neural network, and $S(\bullet)$ is defined by

$$S(\varsigma) = \frac{1}{1 + \exp(-\beta\varsigma)}, \quad \beta > 0, \tag{2.10}$$

where ς is any real value variable.

Consider the problem to approximate the general discrete-time nonlinear system (2.1), by the following discrete-time RHONN series–parallel representation [16]:

$$x_i\,(k+1) = w_i^{*\top} z_i(x(k), \varrho(k)) + \epsilon_{z_i}, \quad i = 1, \cdots, n, \tag{2.11}$$

where x_i is the ith plant state, ϵ_{z_i} is a bounded approximation error, which can be reduced by increasing the number of the adjustable weights [16]. Assume that there exists ideal weight vector w_i^* such that $\|\epsilon_{z_i}\|$ can be minimized on a compact set $\Omega_{z_i} \subset \Re^{L_i}$. The ideal weight vector w_i^* is an artificial quantity required for analytical purpose [16]. In general, it is assumed that this vector exists and is constant but unknown. Let us define its estimate as w_i and the estimation error as

$$\widetilde{w}_i(k) = w_i^* - w_i(k)\,. \tag{2.12}$$

The estimate w_i is used for stability analysis, which will be discussed later. Since w_i^* is constant, then $\widetilde{w}_i(k+1) - \widetilde{w}_i(k) = w_i(k) - w_i(k+1)$, $\forall k \in 0 \cup \mathbb{Z}^+$.

From (2.7) three possible models can be derived:

- Parallel model

$$\widehat{x}_i(k+1) = w_i^\top z_i(\widehat{x}(k), \varrho(k)), \quad i = 1, \cdots, n \tag{2.13}$$

- Series–Parallel model

$$\widehat{x}_i(k+1) = w_i^\top z_i(x(k), \varrho(k)), \quad i = 1, \cdots, n \tag{2.14}$$

- Feedforward model (HONN)

$$\widehat{x}_i(k) = w_i^\top z_i(\varrho(k)), \quad i = 1, \cdots, n, \tag{2.15}$$

  where $\widehat{x}$ is the NN state vector, x is the plant state vector, and ϱ is the input vector to NN.

2.3 The EKF Training Algorithm

The best well-known training approach for recurrent neural networks (RNN) is the back propagation through time learning [21]. However, it is a first order gradient descent method and hence its learning speed could be very slow [10]. Recently, extended Kalman filter (EKF) based algorithms have been introduced to train neural networks [1,3]. With the EKF based algorithm, the learning convergence is improved [10]. The EKF training of neural networks, both feedforward and recurrent ones, has proven to be reliable and practical for many applications over the past 10 years [3].

It is known that Kalman filtering (KF) estimates the state of a linear system with additive state and output white noises [7,20]. For KF-based neural

network training, the network weights become the states to be estimated. In this case, the error between the neural network output and the measured plant output can be considered as additive white noise. Because of the fact that the neural network mapping is nonlinear, an EKF-type is required (see [18] and references therein).

The training goal is to find the optimal weight values, which minimize the prediction error. The EKF-based training algorithm is described by [7]

$$\begin{aligned} K_i(k) &= P_i(k)H_i(k)\left[R_i(k) + H_i^\top(k)P_i(k)H_i(k)\right]^{-1}, \\ w_i(k+1) &= w_i(k) + \eta_i K_i(k)\left[y(k) - \widehat{y}(k)\right], \\ P_i(k+1) &= P_i(k) - K_i(k)\,H_i^\top(k)P_i(k) + Q_i(k), \end{aligned} \tag{2.16}$$

where $P_i \in \Re^{L_i \times L_i}$ is the prediction error associated covariance matrix, $w_i \in \Re^{L_i}$ is the weight (state) vector, L_i is the total number of neural network weights, $y \in \Re^m$ is the measured output vector, $\widehat{y} \in \Re^m$ is the network output, η_i is a design parameter, $K_i \in \Re^{L_i \times m}$ is the Kalman gain matrix, $Q_i \in \Re^{L_i \times L_i}$ is the state noise associated covariance matrix, $R_i \in \Re^{m \times m}$ is the measurement noise associated covariance matrix, $H_i \in \Re^{L_i \times m}$ is a matrix for which each entry (H_{ij}) is the derivative of one of the neural network output, $(\widehat{y})$, with respect to one neural network weight, (w_{ij}), as follows:

$$H_{ij}(k) = \left[\frac{\partial \widehat{y}(k)}{\partial w_{ij}(k)}\right]_{w_i(k)=\widehat{w}_i(k+1)}, \quad i = 1, \ldots, n \text{ and } \; j = 1, \ldots, L_i. \tag{2.17}$$

Usually P_i, Q_i, and R_i are initialized as diagonal matrices, with entries $P_i(0)$, $Q_i(0)$, and $R_i(0)$, respectively. It is important to note that $H_i(k)$, $K_i(k)$, and $P_i(k)$ for the EKF are bounded [20]. Therefore, there exist constants $\overline{H_i} > 0$, $\overline{K_i} > 0$, and $\overline{P_i} > 0$ such that

$$\begin{aligned} \|H_i(k)\| &\leq \overline{H_i}, \\ \|K_i(k)\| &\leq \overline{K_i}, \\ \|P_i(k)\| &\leq \overline{P_i}. \end{aligned} \tag{2.18}$$

Comment 2.1. The measurement and process noises are typically characterized as zero-mean, white noises with covariances given by $\delta_{k,j}R_i(k)$ and $\delta_{k,j}Q_i(k)$, respectively, with $\delta_{k,j}$ a Kronecker delta function (cero for $k \neq l$ and 1 for $k = l$) [8]. To simplify the notation in this book, the covariances will be represented by their respective associated matrices, $R_i(k)$ and $Q_i(k)$ for the noises and $P_i(k)$ for the prediction error.

2.4 Neural Control

To control a system is to force it to behave in a desired way. How to express this "desired behavior" depends primarily on the task to be solved, but the dynamics of the system, the actuators, the measurement equipment, the available

computational power, etc. influence the formulation of the desired behavior as well. Although the desired behavior obviously is very dependent on the application, the need to express it in mathematical terms suited for practical design of control systems seriously limits the means of expression. At the higher level, it is customary to distinguish two basic types of problems [14]:

Regulation problem. The fundamental desired behavior is to keep the output of the system at a constant level regardless of the disturbances acting on the system.

Tracking problem. The fundamental desired behavior is to force the system output to track a reference trajectory closely.

Neural networks (NN) have become a well-established methodology as exemplified by their applications to identification and control of general nonlinear and complex systems [6]; the use of high order neural networks for modeling and learning has recently increased [19]. Specifically, the problem of designing robust neural controllers for nonlinear systems with uncertainties and disturbances, which guarantees stability and trajectory tracking, has received an increasing attention lately.

Using neural networks, control algorithms can be developed to be robust to uncertainties and modeling errors. The most used NN structures are *Feedforward* networks and *Recurrent* ones [19]. The last type offers a better suited tool to model and control nonlinear systems [15].

The neural control problem can be approached in two different ways:

Direct control system design. "Direct" means that the controller is a neural network. A neural network controller is often advantageous when the real-time platform available prohibits complicated solutions. The implementation is simple while the design and tuning are difficult. With a few exceptions this class of designs is model-based in the sense that a model of the system is required in order to design the controller.

Indirect control system design. This class of designs is always model-based. The idea is to use a neural network to model the system to be controlled; this model is then employed in a more "conventional" controller design. The model is typically trained in advanced, but the controller is designed online. As it will appear, the indirect design is very flexible; thus it is the most appropriate for most of the common control problems.

The increasing use of NN to modeling and control is in great part due to the following features that makes them particularly attractive [4]:

- NN are universal approximators. It has been proven that any continuous nonlinear function can be approximated arbitrarily well over a compact set by a multilayer neural network, which consist of one or more hidden layers [2].
- Learning and adaptation. The intelligence of neural networks comes from their generalization ability with respect to unknown data. Online adaptation of the weights is possible.

http://www.springer.com/978-3-540-78288-9

Discrete-Time High Order Neural Control
Trained with Kalman Filtering
Sanchez, E.N.; Alanís, A.Y.; Loukianov, A.G.
2008, X, 110 p., Hardcover
ISBN: 978-3-540-78288-9